

Рубен Ахаян

®
www.bhv.ru
www.bhv.kiev.ua

Macromedia

ColdFusion

- 
- Сервер Web-приложений ColdFusion Server
 - Среда разработки ColdFusion Studio
 - Языки CFML и CFScript

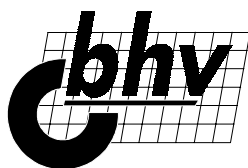
**Наиболее
полное
руководство**

В ПОДЛИННИКЕ®

Рубен Ахаян

Macromedia

ColdFusion



Санкт-Петербург

Дюссельдорф ♦ Киев ♦ Москва ♦ Санкт-Петербург

Книга посвящена Web-проектированию и раскрывает возможности полноценной, высокопроизводительной и легко расширяемой платформы Macromedia ColdFusion, предназначенной для построения крупномасштабных систем электронной коммерции. Большинство глав включает в себя, помимо теории, практический курс использования тегов и функций ColdFusion Markup Language (CFML) и сценариев CFScript. В книге описаны основные шаги по установке и настройке программного продукта, сведения для получения начальных навыков работы в среде ColdFusion Studio, этапы проектирования Web-приложения, включающие постановку задачи, определение бизнес-правил, разработку баз данных и дизайна проекта. Рассматриваются возможности по управлению приложением с помощью конфигурационных файлов, обработка данных с кратким описанием синтаксиса языка SQL, построение диаграмм, управление файлами и каталогами, организация поисковой системы, планирование и генерация статических страниц, взаимодействие с почтовым сервером. Особое внимание уделено технологиям WAP, COM, CORBA, EJB, WDDX.

Для Web-разработчиков

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зав. редакцией	<i>Наталья Таркова</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натали Смирновой</i>
Корректор	<i>Наталия Першакова</i>
Дизайн обложки	<i>Игоря Цырульников</i>
Зав. производством	<i>Николай Тверских</i>

Ахаян Р. А.

Macromedia ColdFusion. — СПб.: БХВ-Петербург, 2002. — 672 с.: ил.

ISBN 5-94157-094-5

© Р. А. Ахаян, 2002

© Оформление, издательство "БХВ-Петербург", 2002

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.10.01.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 54,18.

Тираж 5000 экз. Заказ №

"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99 от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.

Содержание

Предисловие	1
Для кого предназначена эта книга?	2
Структура книги	3
Благодарности.....	3
Введение.....	5
Из истории ColdFusion, или от Allaire к Macromedia	5
Что делает ColdFusion конкурентно-способным?	6
Возможности ColdFusion Server.....	7
Возможности ColdFusion Studio	8
Глава 1. Установка и настройка ColdFusion Server 5	11
Требования к операционной системе	11
Microsoft Windows.....	11
Solaris	13
Linux.....	14
HP-UX.....	14
Установка ColdFusion Server 5 в Windows	15
Процесс установки	17
Конфигурирование Apache для ColdFusion	20
Службы ColdFusion в Windows	21
Учетные записи	21
Настройка ColdFusion Server.....	22
Управление источниками данных	24
Глава 2. Установка и настройка ColdFusion Studio 5	27
Системные требования	27
Установка ColdFusion Studio.....	27
Настройки ColdFusion Studio.....	29
Перекодирование документов.....	30
Шрифты и сворачиваемый текст.....	32
Автоматическая вставка тегов	33
Цветовая схема элементов кодирования.....	35
Шаблоны кодов	36
Автоматическое резервирование.....	37
Проверка правильности написания страниц	38
Правописание	38
Автоматизация процесса формирования кода	40
Настройки <i>General Settings</i>	42
Настройки <i>Tag-specific Settings</i>	43
Настройки <i>HTML Tidy CodeSweeper</i>	45

В дополнение	46
Установка соединения с удаленным сервером	46
FTP-сервер	47
RDS-сервер	48
Глава 3. Интерфейс ColdFusion Studio	51
Панели инструментов	52
Полоса быстрого доступа к наборам инструментов	54
Режимы работы с документами	66
Панель инструментов режима редактирования документов <i>Edit</i>	67
Панель инструментов режима просмотра документов <i>Browse</i>	69
Режим проектирования <i>Design</i>	69
Навигация по открытым документам	70
Панель ресурсов	71
Окно вывода результатов	72
Панель отладки документов	73
Клавиатурные комбинации для быстрого доступа	74
Глава 4. ColdFusion Studio	79
Справочная система	79
Поиск	80
Пользовательские файлы помощи	81
Закладки	82
Контекстная помощь	84
Кнопки помощи в диалоговых окнах	85
Управление файлами и каталогами	85
Работа с каталогами	86
Работа с файлами	86
Работа со списком избранных каталогов	87
Работа с удаленными серверами	88
Загрузка Web-страниц	90
Работа с проектами	91
Создание нового проекта	91
Каталоги проекта	93
Файлы проекта	95
Файлы каталогов ручного вложения и виртуальных каталогов	96
Файлы в каталогах автоматического вложения	96
Ресурсы проекта	96
Использование инспектора тегов	99
Дерево тегов	99
Инспектор тегов	101
Обозреватель сайта	102
Множественно повторяющиеся фрагменты кода	104
Шаблоны кодов	106
Базы данных	107
Просмотр данных	109

Построение запроса	109
Использование запросов	111
Отладка Web-страниц	113
Редактор стилей.....	117
Редактор Image Map.....	118
Работа с буфером обмена	121
Восстановление резервных копий.....	122
Глава 5. Проектирование Web-приложения	125
Постановка задачи	125
Бизнес-правила.....	126
Разработка базы данных.....	127
Физическое описание модели.....	131
Таблицы (Tables).....	132
Связи (References).....	137
Триггеры (Triggers).....	138
Дизайн проекта.....	143
Глава 6. Первая страница с использованием CFML	147
Пишем код CFML-страницы.....	147
Сохранение CFML-страниц.....	148
Просмотр CFML-страниц	148
Анализ кода.....	151
Усложнение кода.....	152
Резюме	160
Глава 7. Cookies в CFML	161
Тег <CFCOOKIE>.....	161
Пример использования тега <CFCOOKIE>	163
Описание кода.....	171
Резюме.....	175
Глава 8. Операторы CFML	177
Арифметические операторы.....	177
Операторы отношения или сравнения	178
Булевы операторы	179
Строковый оператор.....	180
Приоритеты операторов	180
Глава 9. Переменные и типы CFML	183
Типы данных.....	183
Переменные	186
Определение переменных	187
Тег <CFSET>.....	187
Тег <CFPARAM>.....	189
CGI-переменные	190

Клиентские переменные	192
Переменные сессий	195
Переменные прикладных задач	198
Серверные переменные	199
Просмотр переменных	200
Проверка переменных	203
Резюме	209
Глава 10. Управление приложением	211
Планирование структуры приложения	211
Пример файла Application.cfm	213
Тег <CFINCLUDE>	216
Механизм хранения клиентских переменных	216
Хранение клиентских переменных в базе данных	217
Использование блокировок <CFLOCK>	222
Глава 11. Работа с данными	225
Тег <CFQUERY>	226
Команда <i>SELECT</i>	229
Первый запрос	233
Добавление данных	238
Команда <i>INSERT</i>	239
Пример команды <i>INSERT</i>	239
Тег <CFINSERT>	246
Пример тега <CFINSERT>	249
Вывод результатов	255
Тег <CFOUTPUT>	255
Пример тега <CFOUTPUT>	256
Тег <CFTABLE>	261
Тег <CFCOL>	262
Пример сочетания тегов <CFTABLE> и <CFCOL>	263
Модификация данных	264
Команда <i>UPDATE</i>	264
Пример команды <i>UPDATE</i>	265
Тег <CFUPDATE>	271
Пример тега <CFUPDATE>	271
Удаление данных	277
Команда <i>DELETE</i>	277
Пример команды <i>DELETE</i>	277
Транзакции	281
Тег <CFTRANSACTION>	281
Пример тега <CFTRANSACTION>	282
Резюме	283
Глава 12. Построение форм	285
Тег <CFFORM>	286
Тег <CFINPUT>	287

Контроль ввода данных	290
Контроль с помощью регулярных выражений	290
Пример использования атрибута <i>VALIDATE</i>	292
Контроль с помощью кода JavaScript	293
Пример с использованием JavaScript	294
Тег <i><CFTEXTINPUT></i>	297
Тег <i><CFSELECT></i>	299
Тег <i><CFSLIDER></i>	302
Тег <i><CFTREE></i>	306
Тег <i><CFGRID></i>	313
Тег <i><CFAPPLET></i>	326
Резюме	329
Глава 13. Массивы, структуры и списки CFML	331
Массивы	331
Создание массива	331
Проверка массива	332
Добавление элементов	332
Регулирование положением элементов массива	334
Удаление элементов	335
Обработка значений элементов	336
Преобразование массивов	337
Размер массива	337
Структуры	338
Создание структуры	338
Проверка структуры	339
Добавление элементов и модификация структуры	339
Удаление элементов структуры	340
Преобразование структур	341
Другие функции обработки структур	341
Списки	342
Поиск, сортировка, проверка и просмотр списков	342
Изменение и преобразование списков	344
Резюме	346
Глава 14. Разработка пользовательских тегов	347
Создание CF-тегов	347
Тег <i><CFMODULE></i>	356
Создание CFX-тегов	357
Резюме	364
Глава 15. Обработка ошибок	365
Отладка приложения	365
Обработка ошибок	367
Тег <i><CFERROR></i>	367
Типы исключительных ситуаций	368

Создание собственных шаблонов сообщений	369
Обработка исключительных ситуаций	375
Теги <CFTRY>/<CFCATCH>.....	375
Тег <CFTHROW>.....	377
Резюме.....	380
Глава 16. Рассылка и получение почты.....	381
Тег <CFMAIL>.....	381
Рассылка почты	383
Тег <CFPOP>.....	387
Получение почты	389
Пример реализации почтового ящика	392
С чего начать?.....	392
Регистрация.....	394
Шифрование пользовательских паролей	396
Адресная книга.....	398
Создание и отправление почтовых сообщений	401
Просмотр папок с письмами	402
Заключение по реализации почтового ящика.....	407
Резюме.....	408
Глава 17. Управление файлами и каталогами.....	409
Тег <CFFILE>.....	409
Примеры управления файлами.....	413
Тег <CFDIRECTORY>.....	415
Примеры управления каталогами.....	416
Резюме.....	417
Глава 18. Протоколы для работы с удаленными серверами.....	419
Протокол передачи гипертекста.....	419
Тег <CFHTTP>.....	419
Тег <CFHTTPPARAM>.....	422
Примеры работы с удаленным сервером через HTTP-протокол.....	423
Протокол передачи файлов.....	427
Тег <CFFTP> для открытия и закрытия соединения.....	428
Тег <CFFTP> для управления файловой системы FTP-сервера	430
Пример реализации упрощенного FTP-клиента	433
Резюме.....	445
Глава 19. Организация поисковой системы	447
Поисковый сервер Verity.....	447
Принцип организации поисковой системы	448
Управление коллекциями.....	450
Тег <CFCOLLECTION>.....	452
Пример управления коллекциями.....	454
Индексирование данных	454

Тег <CFINDEX>.....	454
Примеры индексирования данных.....	457
Поиск данных.....	458
Тег <CFSEARCH>.....	458
Пример поиска.....	460
Построение выражений поиска.....	463
Использование Verity Server K2.....	466
Резюме.....	469
Глава 20. Язык CFScript	471
Тег <CFSCRIPT>	471
Общие сведения о языке CFScript	472
Глава 21. Тег <CFOBJECT> и технологии COM, CORBA и EJB	481
COM	481
CORBA	482
EJB и Java.....	482
Тег <CFOBJECT>.....	483
Примеры вызова распределенных объектов	486
Java-объекты.....	486
COM-объекты	488
Резюме.....	497
Глава 22. Использование WML-страниц для разработки WAP-приложений.....	499
Подготовка к работе	501
Конфигурация типов MIME для Web-сервера	501
Эмуляторы WAP-браузера.....	502
Обзор языка WML	502
Совместное использование WML и CFML.....	511
Создание диаграммы с помощью тега <CFGGRAPH> посредством мобильного устройства.....	513
Резюме.....	518
Глава 23. Безопасность приложения	519
Безопасность Remote Development Services.....	519
Пользовательская безопасность.....	520
Реализация расширенной безопасности на страницах приложения.....	521
Кодирование страниц приложения.....	522
Функции идентификации и авторизации.....	522
Функция <i>IsAuthenticated</i>	522
Функция <i>IsAuthorized</i>	522
Тег <CFAUTHENTICATE>.....	524
Тег <CFIMPERSONATE>.....	525
Резюме.....	527

Глава 24. Технология Web Distributed Data Exchange	529
Распределение и совместимость платформ	530
Тег <CFWDDX>	530
Пример использования WDDX	531
Языковая поддержка WDDX	533
Глава 25. Настройка ColdFusion-расширений для Macromedia Dreamweaver	535
Использование расширения ColdFusion	536
Приложение 1. Коды основных файлов проекта "My-Line Cruises"	539
Приложение 2. Коды основных файлов проекта "MyMail"	561
Приложение 3. Функции CFML	625
Алфавитный порядок	625
Категории функций	631
Математические функции	631
Символьные функции	633
Системные функции	636
Функции даты и времени	637
Функции идентификации	638
Функции преобразования	639
Функции проверки	639
Функции управления запросами	641
Функции управления массивами	641
Функции управления списками	642
Функции управления структурами	644
Функции форматирования	645
Другие функции	646
Приложение 4. Теги CFML	649
Алфавитный порядок	649
Категории тегов	652
Теги построения форм	652
Теги управления базами данных	653
Теги управления файлами и каталогами	654
Теги управления выводом	654
Теги обработки ошибок	654
Теги расширения функциональности	655
Теги контроля за процессом исполнения страниц	656
Теги Internet-протоколов	656
Теги манипулирования переменными	657
Теги управления приложениями	657
Другие теги	658

Предисловие

Наверное, уже нет необходимости объяснять, что такое Internet или Web-технологии. Сегодня большинство компаний, будь то крупная корпорация или фирма с малым бизнесом, используют информационные технологии для автоматизации своих бизнес-процессов, где Web зачастую выступает в качестве основной технологии, а то и единственной.

Web-технология развивается исключительно быстрыми темпами, а большинство проблем решается совместными усилиями мирового компьютерного сообщества.

В чем же заключается феномен Web-технологий, с одной стороны, обусловивших столь стремительный прорыв массового пользователя к Internet, а с другой стороны, вызвавших гигантский и все нарастающий интерес мировой элиты бизнеса к новым принципам управления информацией, характерным для систем intranet? По мнению многих экспертов, причина состоит в том, что Web-технология, во-первых, опирается на наиболее естественный для человека способ потребления необходимой ему информации (доставка информации по инициативе потребителя), во-вторых, предоставляет универсальный, естественный, интуитивно ясный инструмент для доступа к информации обычного человека, и, в-третьих, является наиболее универсальным подходом к интеграции информационных ресурсов.

"Блуждая" по различным ресурсам глобальной сети Internet, мы можем только догадываться о способе формирования той или иной HTML-страницы. Это может быть как страница, время от времени обновляемая Web-мастером посредством FTP-протокола, так и страница, обновляемая динамически, т. е. непосредственно в теле страницы записывается код, выполняемый в контексте сервера. Например, на Web-сервере размещается база данных и соответствующие "интеллектуальные" страницы с указанием источника данных и включенным запросом к этим данным. На их основе клиент (Web-браузер) отображает совокупность определенной части самой страницы (шаблона) и результата запроса.

Впрочем, не исключен промежуточный вариант, когда на локальном сервере той или иной фирмы страницы создаются динамически, используя корпоративную базу данных, а затем результат отправляется по тому же FTP-протоколу в соответствии с расписанием, указанным все тем же Web-мастером. Для решения подобных задач предназначены программные инструменты, именуемые серверами Web-приложений.

На сегодняшний день существует целый ряд технологий, позволяющих выполнять программы-сценарии на сервере, получая при этом через программы-клиенты результат, отображаемый Web-браузером. Это и Microsoft Active

Server Pages (ASP), и Java Server Pages (JSP), и Perl, и Personal Home Page (PHP), и Sybase Power Dynamo и т. д.

Однако существует еще один программный продукт от корпорации Macromedia. Это ColdFusion. О чем, собственно говоря, и пойдет речь в этой книге.¹

Цель книги — исследование возможностей ColdFusion как одного из средств разработки Web-приложений.

Для кого предназначена эта книга?

ColdFusion ориентирован на серьезных разработчиков, которые создают приложения для Web, поддерживающие электронную коммерцию и базы данных. В программе реализовано визуальное программирование профессионального уровня, а также инструменты разработки баз данных и отладки.

Вы можете создавать сложные SQL-операторы с помощью инструментального набора Visual Database, использовать преимущество встроенного редактора каскадных таблиц стилей, а также отлаживать программы в интерактивном режиме.

Одной из сильных сторон программы является ее способность манипулировать большими проектами, в которых задействовано много разработчиков и серверов. В программу также встроен мощный серверный язык сценариев (ColdFusion Markup Language, CFML) с синтаксисом на основе тегов, который работает с HTML и XML.

ColdFusion обладает поддержкой WML (Wireless Markup Language, язык разметки для беспроводных устройств).

Если вы профессиональный Web-мастер, имеющий в своем арсенале базовые знания по ASP, PHP или Perl, то эта книга поможет вам разобраться еще с одним инструментом разработки Web-сайтов.

Если вы создаете Web-страницы с помощью JavaScript или являетесь студентом, специализирующимся в области Web-технологий, то книга также будет вам полезна.

Если вам пока не доводилось создавать серьезные Web-приложения, но при этом вы имеете некоторый опыт применения HTML, то можете использовать ColdFusion для усовершенствования собственных Web-страниц и в дальнейшем вступить в ряды обладателей почетной профессии — Web-

¹ Надо отметить, что автору этого бестселлера (хочется на это надеяться) довелось совершать вечерние пробежки неподалеку от центрального офиса корпорации Allaire в Бостоне в те времена, когда ColdFusion еще являлся программным продуктом этой компании.

мастер. Однако не обольщайтесь: для того чтобы действительно называться "мастером", надо немало потрудиться.

Структура книги

Посвященная Web-проектированию, эта книга раскрывает возможности полноценной, высокопроизводительной и легко расширяемой платформы Macromedia ColdFusion, предназначенной для построения крупномасштабных систем электронной коммерции.

Для упрощения восприятия большинство глав включает в себя помимо теории практический курс по использованию тегов и функций ColdFusion Markup Language (CFML) применительно к тематике данной главы.

В книге описаны основные шаги по установке и настройке программного продукта, начальные навыки работы в среде ColdFusion Studio, этапы проектирования Web-приложения, включающие постановку задачи, определение бизнес-правил, разработку базы данных и оформление дизайна проекта.

Также в книге рассмотрены основы языка CFML, возможности по управлению приложением с использованием конфигурационных файлов, работа с данными с предоставлением синтаксиса SQL-языка, управление файлами, организация поисковой системы по сайту.

А также в книге описано использование языка CFScript, почтового сервера, COM- и CORBA-объектов, WML-страниц для разработки WAP-приложений.

В конце книги приведены приложения с дополнительной справочной информацией по синтаксису функций и тегов ColdFusion.

Благодарности

Автор выражает искреннюю благодарность школьному учителю английского языка Даниилу Андреевичу Апрелькову, директору школы Льву Григорьевичу Ромашову, в том числе учителю физической культуры Лосю Иосифу Моисеевичу за прекрасное отношение в годы формирования личных качеств автора. Преподавателям Евгению Ивановичу Зайцеву и Андрею Эдливичу Гореву за первые уроки по компьютерной грамматике, коллективу НИПИ территориального развития и транспортной инфраструктуры во главе с Александром Ивановичем Солодким. А также Ирине Буренковой и Игорю Николаевичу Зуеву. А главное, низкий поклон моей покойной маме — Анне Андрониковне, которая неожиданно умерла незадолго до окончания работы автора над этой книгой (как сейчас помню, 2 сентября).

Введение

Macromedia ColdFusion — полноценная, высокопроизводительная, легко расширяемая платформа, предназначенная для разработки Web-приложений, в т. ч. построения крупномасштабных систем электронной коммерции. Она содержит собственную среду выполнения программ, предусматривает поддержку популярных серверных технологий, предоставляет расширенные функции управления и обеспечения безопасности, к тому же на практике имеет большой спрос среди сообщества Web-разработчиков.

ColdFusion, с недавних пор ставший продуктом корпорации Macromedia, сочетает в себе как сервер Web-приложений, ColdFusion Server, так и средство разработки приложений на его платформе, ColdFusion Studio.

Из истории ColdFusion, или от Allaire к Macromedia

Джереми Эллэйр (Jeremy Allaire) с группой программистов основал свою фирму осенью 1994 года. На первом этапе в ней работало двенадцать человек: четверо разработчиков, трое занимались поддержкой пользователей, остальные представляли отделы менеджмента и продаж. К 2001 году количество сотрудников составило порядка пятисот человек.

16 января 2001 года две известные компании Allaire и Macromedia, успешно работающие на рынке программного обеспечения, ориентированного на создание платформ электронного бизнеса, Web-базируемых приложений и творческого оформления Web-сайтов, объявили о слиянии и намерении в дальнейшем вести общий бизнес.

Если быть более точными, то это объединение на самом деле является выгодным приобретением Macromedia компании Allaire. За каждую акцию Allaire Macromedia отдал 0,2 собственных акций и еще 3 доллара. Таким образом, сумма сделки составляла около 360 млн долларов. Следовательно, объединенная компания носит название Macromedia, имеет штаб-квартиру в Сан-Франциско и офис в Бостоне. Главой компании на период написания книги оставался Роб Бергесс (Rob Burgess), председатель правления CEO Macromedia, а Джереми Аллэйр (Jeremy Allaire) занял пост технического директора (Chief Technical Officer, CTO), этот же пост он занимал и в компании Allaire.

До подписания договора о слиянии компании продолжали работать как самостоятельные структуры, независимо друг от друга. После прохождения всех формальностей оформления сделки Allaire и Macromedia объединили

свои партнерские каналы и с конца второго квартала 2001 года осуществляют совместную поддержку всех своих продуктов.

Итак, в мире появилась компания, которая способна предоставить своим пользователям практически весь спектр программных решений для обеспечения жизненного цикла Web-серверов любой сложности, от простейших домашних страничек до сложнейших порталов электронной коммерции. До объединения компании Allaire и Macromedia вели достаточно тесное сотрудничество в области Web-технологий и прикладывали все возможные усилия для взаимной интеграции своих продуктов так, что дальнейшее объединение линеек программного обеспечения не должно вызвать больших трудностей. Как говорится в совместном пресс-релизе компаний, основная ставка будет сделана на такие продукты, как Allaire ColdFusion, Allaire JRun, Macromedia Dreamweaver и Macromedia Flash. В принципе, эти продукты как раз и обеспечивают все стадии разработки и поддержки Web-приложений.

Что делает ColdFusion конкурентно-способным?

Согласно различным исследованиям, ColdFusion — в США один из лидеров по продажам среди Web-серверных приложений. Однако нельзя забывать о существовании ASP, JSP, PHP, Perl и подобных конкурирующих технологиях. Тем более, что большинство из вышеперечисленных являются свободно распространяемыми. При этом большое количество людей используют ColdFusion. Можно выделить три основные причины, по которым стоит выбрать ColdFusion.

Первая причина — *простота*. Многие разработчики выбирают ColdFusion, исходя из простоты обучения. Большинство разработчиков с HTML-навыками находят язык ColdFusion, который часто упоминается как CFML (ColdFusion Markup Language, язык разметки ColdFusion), весьма интуитивно понятным.

Вторая причина — *мощность*. Несмотря на факт, что ColdFusion является простым в изучении, это также чрезвычайно мощное средство разработки. Имеется более семидесяти тегов, более двухсот сорока функций на языке CFML. Краткий обзор главных услуг, предлагаемых через CFML, таков:

- ☐ работа с данными через ODBC, OLE DB и собственные драйверы для Oracle, Sybase, Informix, DB2;
- ☐ работа с электронной почтой через протоколы POP и SMTP;
- ☐ работа с каталогами через протокол LDAP;
- ☐ доступ к файловым системам напрямую с помощью драйверов и через протокол FTP;

- ☐ доступ к распределенным объектам с помощью технологий COM, CORBA и EJB;
- ☐ разработка пользовательских тегов, используя CFML, C++, Java или Delphi;
- ☐ поддержка модели обмена данными WDDX (Web Distributed Date Exchange);
- ☐ безопасность;
- ☐ контроль управления потоками.

Третья причина — *сообщество*. Одна из существенных причин для выбора ColdFusion — сообщество ColdFusion-разработчиков. Наличие открытых исходных кодов является причиной создания общин разработчиков-активистов, которые сосредотачиваются на специфическом программном продукте или инструменте. Подобная группа сформировалась и вокруг ColdFusion. От активных форумов разработчиков на страницах Macromedia&Allaire к активным ColdFusion-группам пользователей (CFUGs) во всем мире. Всегда кто-то найдется, чтобы помочь вам с решением возникшей проблемы. Активное сообщество предоставляет сотни пользовательских тегов для ColdFusion, множества полезных примеров, различных публикаций.

Возможности ColdFusion Server

- ☐ *Поддерживаемые серверы*. ColdFusion Server может работать с любыми Web-серверами Microsoft Personal Web Server, Microsoft IIS, Netscape Enterprise Server и Apache, O'Reilly WebSite Professional.
- ☐ *Интеграция широкого спектра технологий*, включая работу с данными через ODBC, OLE DB и собственные драйверы для Oracle, Sybase, Informix, DB2. Также обеспечивается работа с почтой через протоколы POP и SMTP, каталогами через протокол LDAP, доступ к файловым системам напрямую с помощью драйверов и через протокол FTP, поддержка модели обмена данными WDDX, доступ к распределенным объектам с помощью технологий COM, CORBA и EJB, поддержка протокола SNMP для реализации управления компонентами программы посредством сети.
- ☐ *Удобство администрирования*. ColdFusion Server предоставляет широкий спектр функций администрирования, включая Web-интерфейс, разделение функций администрирования между несколькими пользователями, мониторинг производительности, возможность детальной отладки всех аспектов работы создаваемых Web-приложений, а также развертывание, архивация и восстановление приложений, использующих файлы архивов ColdFusion.
- ☐ *Восстановление после сбоя служб сервера*. Кластеры ColdFusion обеспечивают возможность мониторинга каждого из серверов ColdFusion, поддер-

живают автоматическое обнаружение проблем в работе программного или аппаратного обеспечения с адекватной реакцией в виде автоматической переадресации пользователей на другие серверы и перезапуска служб на проблемном сервере.

- ❑ *Диагностика неисправностей.* Обеспечивается отслеживание неисправностей сервера со встроенным составлением отчетов и формированием протокола сообщений в log-файлах. Существует возможность просмотра (view), разгрузки (download), резервирования (store) и удаления (delete) log-файлов.
- ❑ *Обеспечение защиты.* Интегрированная система аутентификации позволяет использовать существующие службы безопасности каталогов LDAP или доменов Windows NT. Поддержка SSL-соединений, сертификатов X.509, шифрования страниц приложений, а также двухуровневый защитный барьер для исполняемых приложений на уровне операционной системы и на уровне сервера обеспечивают высочайшую степень безопасности в условиях агрессивной окружающей сетевой среды.

Возможности ColdFusion Studio

- ❑ *Удобство расширения.* Программируемая объектная модель визуальных инструментов VTOM (Visual Tool Object Model) обеспечивает возможность создания макросов и расширений среды разработки с использованием языков JavaScript и VBScript, а также ActiveX- или COM-компонентов. Для создания новых визуальных редакторов тегов и мастеров можно использовать язык VTML (Visual Tool Markup Language) и специальный XML-словарь.
- ❑ *Улучшенная совместимость с базами данных.* Для выборки, вставки, обновления или удаления данных из любой ODBC-базы достаточно одной команды CFML. Собственные драйверы предоставляют прямой доступ к базам данных Oracle, Sybase, Informix, DB2. Специальный драйвер обеспечивает работу с OLE DB-источниками, такими как MS Exchange. Интеграция данных из гетерогенных источников, объединяя результаты запросов к данным в памяти с использованием стандартного языка SQL.
- ❑ *Построение интеллектуальных поисковых приложений.* Средство полнотекстового и индексного поиска Verity K2 обеспечивает прозрачную индексацию и эффективный поиск неструктурированных текстовых данных, расположенных в реляционных базах данных, а также в текстовых или других офисных документах. При этом выполняется высокопроизводительный поиск с одновременной поддержкой до 250 000 документов.
- ❑ *Построение отчетов и диаграмм на основе запросов к данным.* Формирование отчетов профессионального уровня для Crystal Reports 8.0, создание

диаграмм различных видов, таких как пространственные "пироги", графики, горизонтальные и вертикальные столбцы с возможностью масштабирования в реальном времени.

- ❑ *Создание функций многократного использования*, позволяющих ускорить процесс разработки приложений различных уровней.
- ❑ *Работа с почтовыми серверами*. Можно динамически создавать и рассылать почтовые сообщения, используя любой SMTP-сервер. За основу можно брать статическую информацию, поля форм или результаты запросов для управления адресами и содержимым почтовых сообщений. Обеспечивается одновременная рассылка сотен сообщений с динамической настройкой их содержимого и применением форматирования обычным текстом или HTML-форматом, получение почтовых сообщений с любых POP-серверов с интеграцией обработки этих сообщений в приложениях ColdFusion.



Установка и настройка ColdFusion Server 5

ColdFusion Server — это высокопроизводительный, легко расширяемый сервер Web-приложений, предназначенный для построения крупномасштабных систем электронной коммерции. Собственная среда исполнения программ, поддержка популярных серверных технологий, расширенные функции управления и обеспечения безопасности, а также балансировка нагрузки и автовосстановление при сбоях — все это гарантирует быстрое развертывание масштабируемых приложений на любой платформе.

Требования к операционной системе

ColdFusion Server 5 может работать на таких платформах, как: Windows, Solaris, Linux и HP-UX. Далее опишем требования к системе относительно каждой платформы.

Microsoft Windows

Минимальные требования, необходимые для установки и управления ColdFusion Server 5 для Windows, зависят от типа редакции устанавливаемого продукта — Professional Edition или Enterprise Edition. В табл. 1.1 приведены эти требования в зависимости от выпуска.

Таблица 1.1. Требования к платформе MS Windows

Требование	ColdFusion Server Professional	ColdFusion Server Enterprise
Версия		
Windows 98	✓	
Windows NT 4.0 (Workstation и Server)	✓	✓
Windows 2000 (Professional*, Server и Advanced Server)	✓	✓

Таблица 1.1 (окончание)

Требование	ColdFusion Server Professional	ColdFusion Server Enterprise
Аппаратное обеспечение		
Процессор	Pentium	Pentium
Минимальное ОЗУ (Мбайт)	128	256
Рекомендуемое ОЗУ (Мбайт)	256	512
Свободное пространство на жестком диске (Мбайт)	200	400
Устройство CD-ROM	✓	✓
Web-сервер		
Microsoft Personal Web Server (PWS)	✓	
Microsoft IIS 4.0, 5.0	✓	✓
iPlanet/Netscape Enterprise Server 3.6, 4.1	✓	✓
Apache Server 1.3.6 или более поздняя версия	✓	✓
O'Reilly WebSite Professional 2.0, 3.0	✓	✓
WebSite Server API (WSAPI)	✓	✓

* ClusterCATS и менеджер приложений не поддерживаются на этих платформах.

Как известно, к любой операционной системе по прошествии некоторого времени после выхода очередной версии поставляются исправления, доработки существующих ошибок. В этой связи большинство программных продуктов имеет требования к таким исправлениям, и ColdFusion Server 5 — не исключение. В табл. 1.2 приведены требования к исправлениям в зависимости от версии Windows.

Таблица 1.2. Требования к исправлениям

Операционная система	Обновления
Windows 98	Service Pack 1, MDAC 2.5 SP1, MFC/MSVC 6.0, NT Option Pack
Window NT 4.0	Service Pack 6a, MDAC 2.5 SP1, MFC/MSVC 6.0, Simple Network Management Protocol (SNMP)* для Management Information Base (MIB)
Windows 2000	Service Pack 1, MDAC 2.5 SP1, MFC/MSVC 6.0 SNMP* для MIB

* Только для Windows NT 4.0 Server и Windows 2000 Server

Solaris

Для платформы Solaris ColdFusion Server 5, как и сервер предыдущей версии, реализован только в виде редакции Enterprise Edition. Для установки и благополучного использования ColdFusion Server 5 под управлением Solaris ваша система должна выдерживать минимальные требования, указанные в табл. 1.3.

Таблица 1.3. Требования к платформе Solaris

Требование	ColdFusion Server Enterprise
Версия	
Solaris 2.6, 7, 8	✓
Аппаратное обеспечение	
Процессор	SPARC
Минимальное ОЗУ (Мбайт)	256
Рекомендуемое ОЗУ (Мбайт)	512
Свободное пространство на жестком диске (Мбайт)	350
Устройство CD-ROM	✓
Web-сервер	
iPlanet/Netscape Enterprise Server 3.6, 4.1	✓
Apache Server 1.3.x	✓

Как уже было отмечено, любая операционная система через некоторое время подвергается доработкам и выпуску различных "заплаток" (patch), и большинство программных продуктов предъявляет требования к таким доработкам. Так и для "заплаток" платформы Solaris, ColdFusion Server 5 имеет определенные требования, указанные в табл. 1.4.

Таблица 1.4. Требования к исправлениям

Операционная система	Заплатки
Solaris 2.6	105181-17 или выше, 105591-09 или выше, 105210-25 или выше, 105568-14 или выше, Solstice Enterprise Agent (SEA) Runtime 1.0.3
Solaris 7	106541-08 или выше, 106327-08 или выше, 106980-07 или выше, 107709-10 или выше
Solaris 8	108869-03 или выше, en_US locale

Все версии Solaris требуют пакет утилит SUNWxcu4 — XCU4 Utilities package.

Linux

Минимальные требования, предъявляемые к установке и управлению ColdFusion Server 5 для Linux, зависят от типа редакции устанавливаемого продукта (Professional Edition или Enterprise Edition) и перечислены в табл. 1.5 в зависимости от выпуска, а требования к "заплаткам" — в табл. 1.6.

Таблица 1.5. Требования к платформе Linux

Требование	ColdFusion Server Professional	ColdFusion Server Enterprise
Linux-дистрибутив		
Red Hat Linux 6.2 и позже	✓	✓
SuSE Linux 7.0 и позже	✓	✓
Cobalt RAQ3, RAQ4, XTR	✓	✓
Аппаратное обеспечение		
Процессор	Pentium	Pentium
Минимальное ОЗУ (Мбайт)	128	256
Рекомендуемое ОЗУ (Мбайт)	256	512
Свободное пространство на жестком диске (Мбайт)	200	350
Устройство CD-ROM	✓	✓
Web-сервер		
iPlanet/Netscape Enterprise Server 4.1	✓	✓
Apache Server 1.3.6	✓	✓

Таблица 1.6. Требования к исправлениям

Операционная система	Заплатки
Red Hat 6.2, 7.0	compat-libstdc++ RPM, UCD-SNMP 4.2.1
SuSE 7.0	apache.rpm
Cobalt RAQ3, RAQ4, XTR	glibc-2.1.3-22, glibc-profile-2.1.3-22, glibc-devel-2.1.3-22, Red Hat 6.0 glibc patch

HP-UX

Для платформы HP-UX программный продукт ColdFusion Server 5 разработан только в виде редакции Enterprise Edition. Для установки и благополуч-

ного использования ColdFusion Server 5 под управлением HP-UX ваша система должна выдерживать минимальные требования, указанные в табл. 1.7.

Таблица 1.7. Требования к платформе HP-UX

Требование	ColdFusion Server Enterprise
Версия	
HP-UX 11.0	✓
Аппаратное обеспечение	
Процессор	PA-RISC 1.1, 2.0
Минимальное ОЗУ (Мбайт)	128
Рекомендуемое ОЗУ (Мбайт)	128
Свободное пространство на жестком диске (Мбайт)	120
CD-ROM устройство	✓
Web-сервер	
iPlanet/Netscape Enterprise Server 3.51, 4.1	✓
Apache Server 1.3.x	✓

Перед установкой ColdFusion Server 5 на платформу HP-UX версии 11.0 должны быть сконфигурированы минимальные значения параметров ядра системы, представленные в табл. 1.8.

Таблица 1.8. Требования к параметрам ядра

Параметр	Значение
maxdsize	0x79000000
maxssize	0x80000000
maxtsize	0x80000000
maxfiles	1024
Maxusers	64 (чтобы косвенно повысить NKTHREAD, используемый ниже)
max_thread_proc	(NKTHREAD-10)

Установка ColdFusion Server 5 в Windows

Перед началом инсталляции удостоверьтесь, что на вашем компьютере установлен Web-сервер. Чтобы убедиться в работоспособности Web-сервера, достаточно запустить привычный для вас Web-браузер и в строке адреса URL (Uniform Resource Locator, унифицированный локатор ресурса) набрать и под-

твердить **http://127.0.0.1/index.html** или **http://127.0.0.1/default.htm** или просто **http://127.0.0.1/**, где 127.0.0.1 — адрес локальной машины — localhost.

В последнем варианте Web-браузер загрузит страницу с Web-сервера, указанную в качестве основной страницы сервера. Например, на многих Web-серверах по умолчанию файл основной (домашней) страницы носит имя index, причем его расширение может быть html, htm, asp, php, jsp, или cfm в зависимости от используемой технологии. Однако имя файла основной страницы Web-сервера на вашем компьютере всегда можно изменить.

Если эксперимент не удался, и вы получили сообщение об ошибке, не огорчайтесь. Вполне возможно, что Web-сервер находится в режиме "глубокой спячки", и для его "пробуждения" достаточно воспользоваться утилитой **Services** (Службы) из Control Panel (Панели управления). Выделите используемый Web-сервер и нажмите кнопку **Start** (Запуск). При успешном выполнении данной команды выделенная служба примет значение статуса, равное **Started** (Работает).

Редакция сервера ColdFusion Server 5 Enterprise Edition имеет дополнительные возможности по управлению информацией — Management Information Base (MIB) — и позволяет управлять разнообразными компонентами программы посредством сети. Для реализации данной возможности необходимо предварительно установить простой протокол управления сетью (Simple Network Management Protocol, SNMP), через который и предоставляется данная услуга.

В Windows протокол SNMP можно установить только в Windows NT 4.0 Server и Windows 2000 Server.

Для установки SNMP в Windows NT 4.0 Server выполните следующие шаги:

1. Запустите утилиту **Network** (Сеть) из Control Panel (Панели управления).
2. В диалоговом окне **Network** (Сеть) выберите вкладку **Services** (Службы) и нажмите кнопку **Add** (Добавить) для отображения окна **Select Network Service** (Выбор типа компонента).
3. Выделите в списке пункт **SNMP Service** (Служба SNMP) и нажмите кнопку **OK**.

Завершив процесс копирования необходимых файлов из дистрибутива Windows NT, программа предложит перезагрузить компьютер для последующего применения установок. Надо отметить, что после установки SNMP, как и после установки некоторых других служб, необходимо переустановить текущие исправления к операционной системе из Service Pack.

Для успешной установки ColdFusion Server вы должны обладать правами администратора, т. е. получить доступ к системе либо под именем "administrator" (администратор), либо под любым другим именем, входящим в группу "administrators" (администраторы).

Процесс установки

Установка ColdFusion Server 5 является достаточно простой процедурой. Вы предварительно закрываете все открытые приложения. Вставляете в устройство чтения CD-ROM установочный компакт-диск и, щелкнув на файле `setup.exe`, последовательно отвечаете на поставленные вопросы, начиная с лицензионного соглашения на программный продукт, указания серийного номера и месторасположения программы, и заканчивая щелчком на кнопке **Finish** (Готово).

Итак, после успешного автоконтроля требований к системе программа предоставит возможность продолжить установку нажатием кнопки **Next** (Далее). Затем после соглашения с условием лицензии на Macromedia ColdFusion Server 5 и подтверждением пользовательской информации с указанием серийного номера необходимо выбрать Web-сервер, как показано на рис. 1.1. Если у вас уже установлен Web-сервер, программа установки ColdFusion Server автоматически это обнаружит.

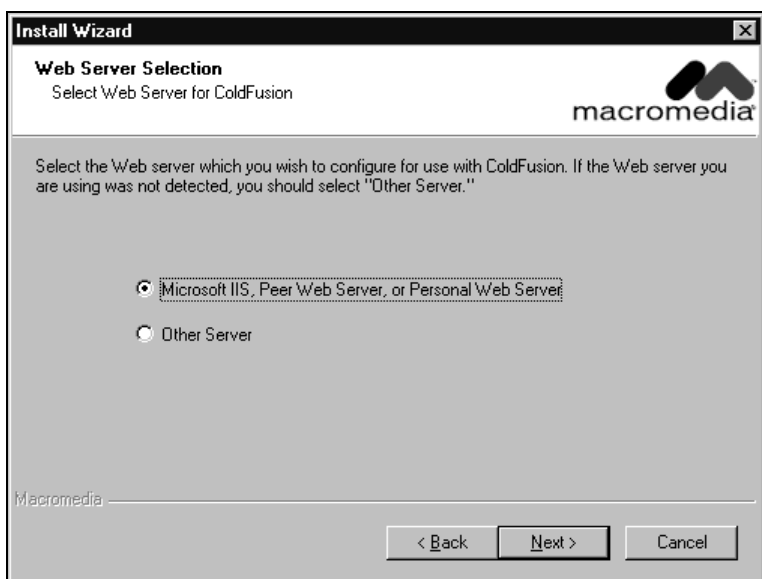


Рис. 1.1. Диалоговое окно выбора Web-сервера

Следующим шагом необходимо указать месторасположения ColdFusion-сервера и корневого каталога Web-сервера (рис. 1.2).

Затем с помощью программы установки можно осуществить выбор необходимых компонентов ColdFusion Server 5, что отражено в диалоговом окне на рис. 1.3.

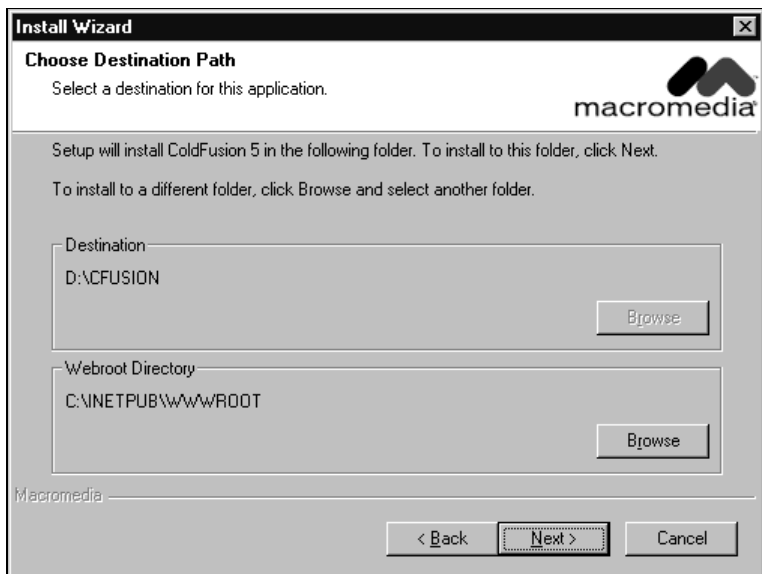


Рис. 1.2. Диалоговое окно этапа указания месторасположения

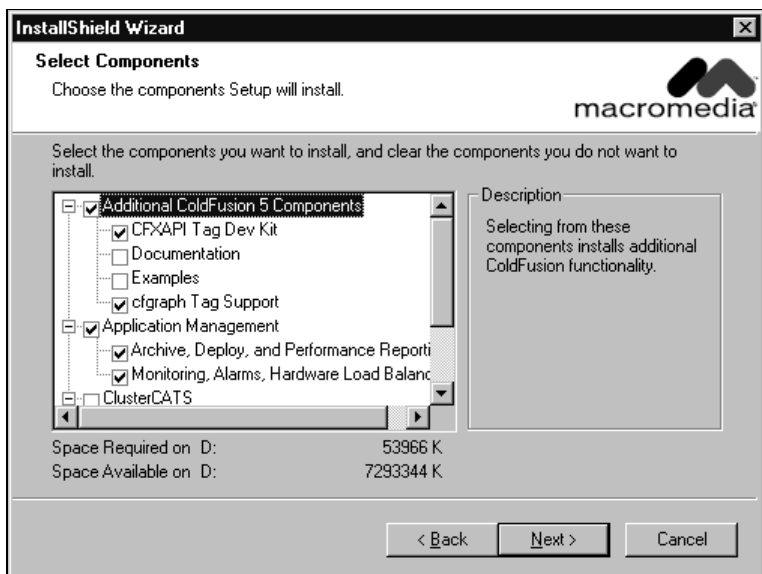


Рис. 1.3. Диалоговое окно списка компонентов ColdFusion Server

В дальнейшем программа установки предложит вам указать пароль администратора ColdFusion Server для доступа к серверной системе с целью выполнения административных работ и пароль пользователя ColdFusion Studio,

как показано на рис. 1.4. После чего, подтвердив все ранее выбранные настройки, начнется процесс копирования ColdFusion Server 5.

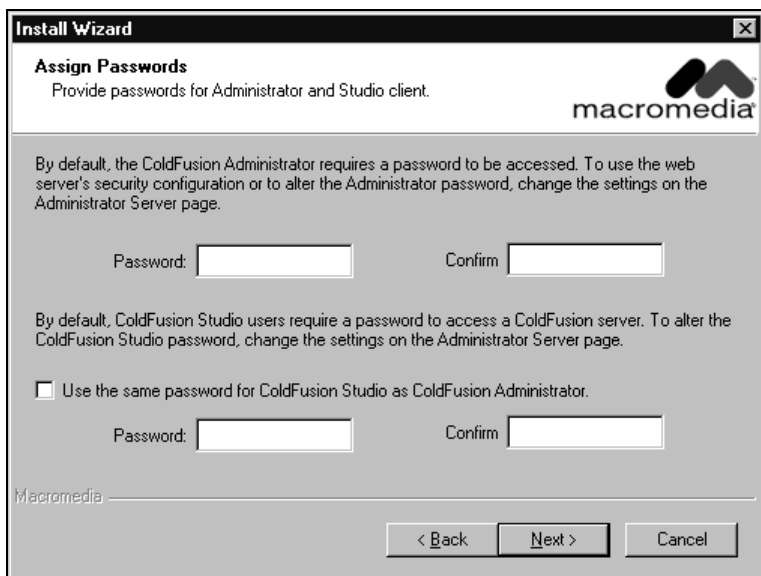


Рис. 1.4. Диалоговое окно указания паролей

Если ваша система имеет предыдущую версию ColdFusion, мастер установки заменяет ColdFusion-файлы программы, но не удаляет существующие страницы Web-приложения, которые могут присутствовать на вашем компьютере.

После окончания копирования файлов и модернизации реестра Windows для ColdFusion программа установки предложит вам завершить процесс инсталляции, нажав кнопку **Finish** (Готово), что и следует сделать.

При удачной установке программы и нахождении Web-сервера в рабочем состоянии, после перезагрузки операционной системы будет произведена попытка открытия Web-страницы "Welcome to ColdFusion 5" с URL-адресом <http://127.0.0.1/CFIDE/administrator/docs/index.cfm>.

Также эту страницу можно открыть, воспользовавшись главным меню Windows и из списка программ выбрав пункт **Macromedia ColdFusion Server 5 | Welcome to ColdFusion**. На рис. 1.5 приведена эта Web-страница.

Данная Web-страница предоставляет возможность зарегистрировать программный продукт, проверить благополучно установленные компоненты, открыть страницу "ColdFusion Administrator", позволяющую администрировать сервер ColdFusion. При этом остается только вспомнить пароль, указанный во время установки сервера.

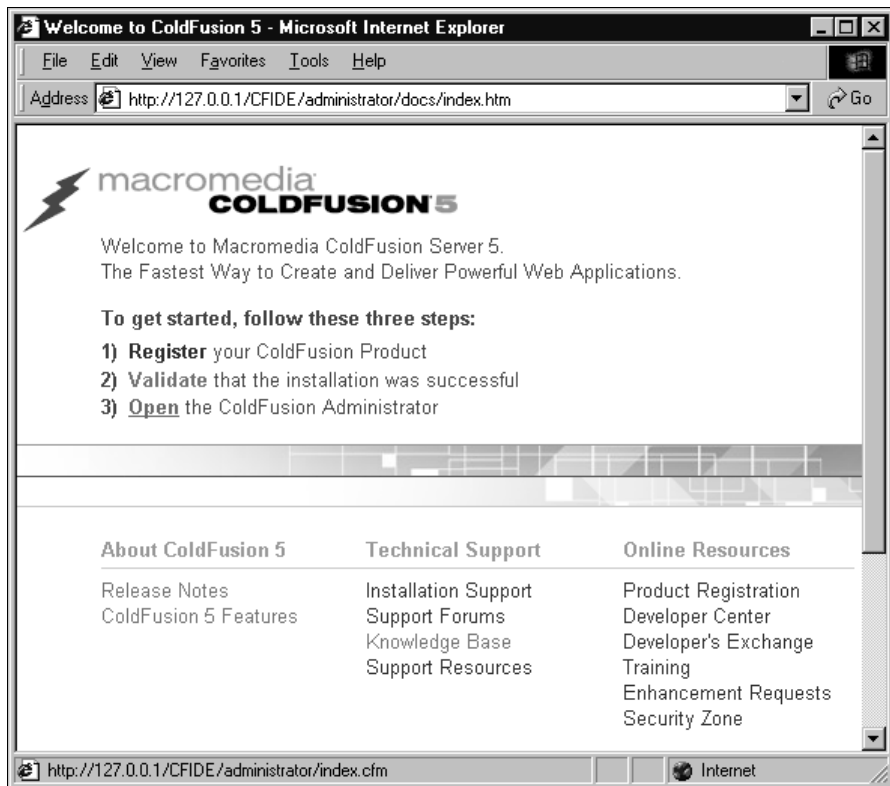


Рис. 1.5. Страница приглашения "Welcome to ColdFusion"

Конфигурирование Apache для ColdFusion

Для конфигурации Web-сервера Apache для ColdFusion достаточно скопировать необходимый модуль в каталог Apache-модулей и зарегистрировать этот модуль в конфигурационном файле Web-сервера Apache. Например, если Apache установлен в c:\Apache, то следует произвести следующие действия:

1. Если Web-сервер Apache запущен, приостановите работу Web-сервера.
2. Скопируйте модуль `ApacheModuleColdFusion.dll` из каталога `\cfusion\bin` в каталог Apache-модулей, например: `c:\Apache\modules\ApacheModuleColdFusion.dll`.
3. Отредактируйте конфигурационный файл (`c:\Apache\conf\httpd.conf`), дополнив его следующей строкой:

```
LoadModule coldfusion_module modules/ApacheModuleColdFusion.dll
```

4. Вновь запустите Apache Web-сервер.

Службы ColdFusion в Windows

После установки ColdFusion Server список сервисов операционной системы MS Windows будет пополнен службами, представленными в табл. 1.9.

Таблица 1.9. Службы ColdFusion в Windows

Служба	Назначение
ColdFusion Application Server	Основная служба ColdFusion Server, без которой ни одна CFML-страница не может быть загружена
ColdFusion Executive	Опрашивает службу ColdFusion Application Server и автоматически перезапускает ее, если эта служба находится в нерабочем состоянии
ColdFusion RDS (Remote Development Service)	Служба, обеспечивающая безопасность каталогов и файлов при просмотре и отладке с помощью ColdFusion Studio
SiteMinder Authentication Service*	Служба для обеспечения повышенной безопасности и проверки подлинности
SiteMinder Authorization Service*	Служба для обеспечения повышенной безопасности и организации доступа
ColdFusion Management Service*	Служба управления ColdFusion, включающая операции архивации/восстановления и сервер составления отчетов
ColdFusion Monitoring Service*	Служба, позволяющая контролировать работоспособность ColdFusion Server, HTTP Server; поддерживает ClusterCATS и мониторинг приложений
ColdFusion Graphing Server*	Обеспечивает выполнение Java runtime для возможности построения диаграмм (CFGGRAPH)
ColdFusion Management Repository Server*	Обеспечивает средства управления хранилищем для службы управления ColdFusion

* Доступно только в ColdFusion Enterprise Edition

Учетные записи

Как любая служба (service) операционной системы, а именно в таком виде выполняется сам ColdFusion и его основные службы при работе под управлением Windows, может работать на основе *локальной учетной записи* (Local system account) или *учетной записи пользователя, зарегистрированного в домене* (Domain user account).

По умолчанию ColdFusion использует локальную учетную запись. Однако при этом страницы приложения ColdFusion не имеют доступа для взаимодействия с удаленными источниками данных, COM-объектами и другими службами,

требующими наличия зарегистрированной учетной записи. Поэтому если вам недостаточно применения ColdFusion в пределах локального компьютера, используйте службу сервисов **Services** для назначения учетной записи пользователя применительно к ColdFusion Application Server (рис. 1.6).

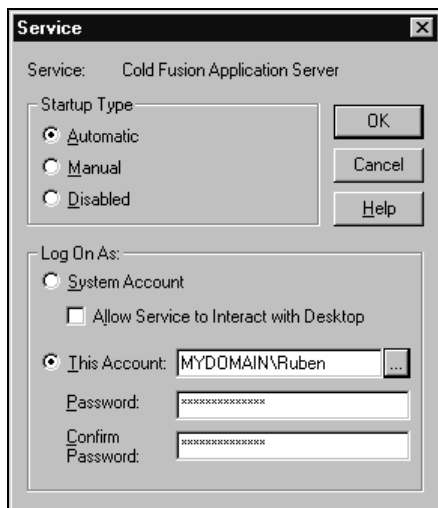


Рис. 1.6. Диалоговое окно параметров службы ColdFusion Application Server

Настройка ColdFusion Server

После успешной установки ColdFusion Server 5 можно приступить к конфигурированию сервера. Для этой цели служит система администрирования, расположенная на Web-странице "ColdFusion Administrator" (рис. 1.7), зайти на которую можно из главного меню Windows, последовательно выбрав **Programs | Macromedia ColdFusion Server 5 | ColdFusion Administrator** (Программы | Macromedia ColdFusion Server 5 | ColdFusion Administrator).

Страница для администрирования ColdFusion Server 5 разделяется на три основных категории: **Server** (Сервер), **Security** (Безопасность) и **Tools** (Инструментарий). В свою очередь каждая категория еще делится на разделы.

Категория **Server** (Сервер) включает следующие разделы:

- ❑ **Server Settings** (Параметры сервера) — позволяет настраивать основные параметры сервера (**Settings**), конфигурировать настройки по кэшированию сервера (**Caching**), определять механизм хранения клиентских переменных (**Client Variables**), управлять жизненным циклом переменных (**Memory Variables**), определять тип блокировок данных для управления синхронным

доступом (**Locking**), настраивать логические псевдонимы (**Mappings**), устанавливать параметры сервера почтовой службы (**Mail/Mail Logging**);

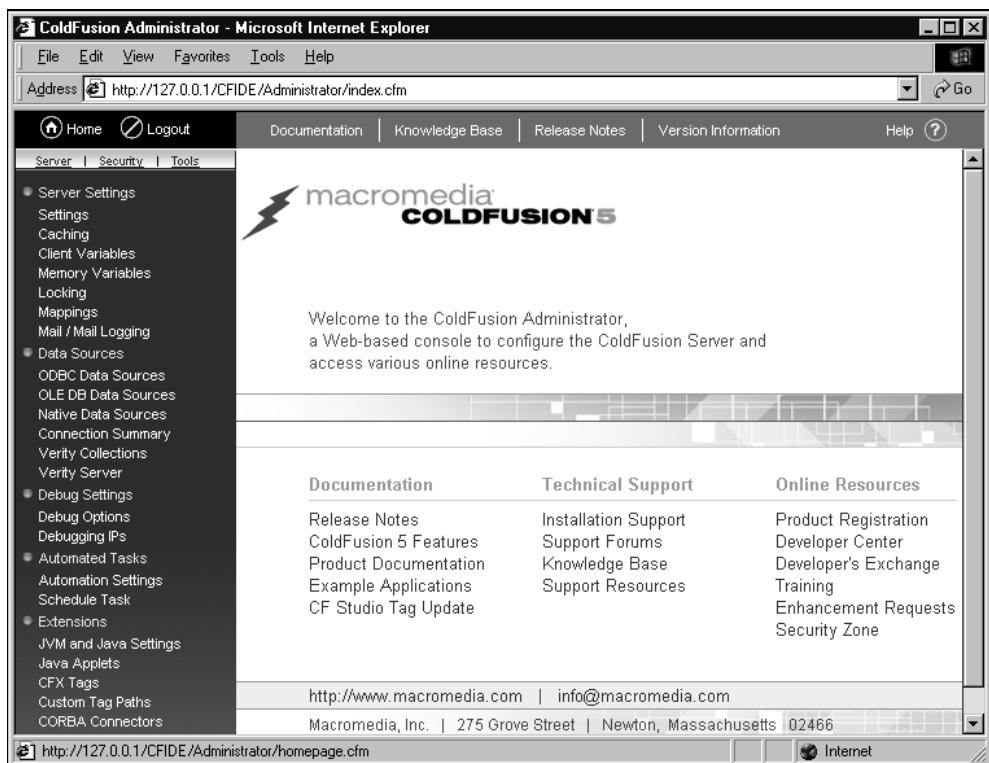


Рис. 1.7. Страница для администрирования ColdFusion Server

- ❑ **Data Sources** (Источники данных) — позволяет конфигурировать источники данных ODBC (**ODBC Data Sources**), конфигурировать источники данных OLE DB (**OLE DB Data Sources**), устанавливать "родные" драйверы баз данных для Oracle, Informix, DB2 и Sybase (**Native Data Sources**), просматривать весь список настроенных соединений с возможностью проверять работоспособность, как по отдельности, так и целиком всех соединений (**Connection Summary**), управлять наборами коллекций полнотекстового и индексного поиска Verity K2 (**Verity Collections**), устанавливать параметры поисковой машины (**Verity Server**);
- ❑ **Debug Settings** (Параметры отладки) — предоставляет возможность конфигурировать параметры отладочного механизма страниц (**Debug Options**), указывать компьютеры для исключения из отладочного процесса (**Debugging IPs**);

- ❑ **Automated Tasks** (Автоматизированные задачи) — позволяет определять периодичность выполнения задач (**Automation Settings**), добавлять новые расписания по выполнению задач (**Schedule Task**);
- ❑ **Extensions** (Расширения) — позволяет устанавливать виртуальную машину Java и конфигурировать параметры, позволяющие ColdFusion работать с Java (**JVM and Java Settings**), регистрировать Java-апплеты (**Java Applets**), регистрировать CFX-теги (**CFX Tags**), определять пути к пользовательским тегам (**Custom Tag Paths**), управлять соединениями к CORBA-объектам (**CORBA Connectors**).

Категория **Security** (Безопасность) содержит следующие разделы:

- ❑ **Basic Security** (Основная безопасность) — позволяет устанавливать пароль для администратора ColdFusion (**CF Admin Password**), указывать ограничения на определенные теги (**Tag Restrictions**), задавать пароль для ColdFusion Studio (**CF Studio Password**);
- ❑ **Advanced Security** (Расширенная безопасность) — дает возможность конфигурировать параметры повышенной безопасности ColdFusion (**Security Configuration**).

Категория **Tools** (Инструментарий) включает разделы:

- ❑ **Logs and Statistics** (Журналы и статистика) — позволяет указывать почтовый адрес администратора и устанавливать параметры протоколирования вплоть до указания каталога для файлов с целью ведения протоколов в ColdFusion (**Logging Settings**), просматривать содержимое файлов-протоколов, а также разгружать, резервировать и удалять эти файлы (**Log Files**), просматривать различные статистические отчеты по работе ColdFusion Server (**Server Reports**);
- ❑ **Archive and Deploy** (Архивация и восстановление) — обеспечивает установку параметров для архивирования (**Archive Settings**), создание (**Create Archive**) и восстановление архивов (**Deploy Archive**), авторизованный доступ к собственным архивам, шифрование и расшифровывание архивов (**Archive Security**).

Автор не преследовал цель детально описать все вышеизложенные установки, да и на первом этапе нет необходимости что-либо изменять в перечисленных настройках. Однако для примера разберем достаточно полезную услугу — управление источниками данных.

Управление источниками данных

Как уже было сказано, с помощью управления источниками данных можно устанавливать "родные" драйверы баз данных для Oracle, Informix, DB2 и Sybase, а так же добавлять драйверы ODBC, OLE DB и проверять работоспособность источников данных. Например, мы создали базу данных с по-

мощью одного из офисных продуктов — Microsoft Access — и назвали эту базу данных Museums.mdb. Для регистрации источника данных необходимо:

1. В разделе **Data Sources** (Источники данных), находящемся в левом фрейме категории **Server** (Сервер), выбрать пункт **ODBC Data Sources** (Источники данных ODBC).
2. В поле **Date Source Name** (Имя источника данных) ввести название нового источника (в нашем примере — Museums).
3. В списке **ODBC Driver** (Драйвер ODBC) выбрать значение **Microsoft Access Driver (*.mdb)**.
4. Нажать кнопку **Add** (Добавить) — рис. 1.8.
5. Указать путь к файлу в поле **Database File** (Файл базы данных); для облегчения поиска можно воспользоваться кнопкой **Browse Server** (Обзор серверов), находящейся напротив этого поля.
6. Нажать кнопку **Create** (Создать).

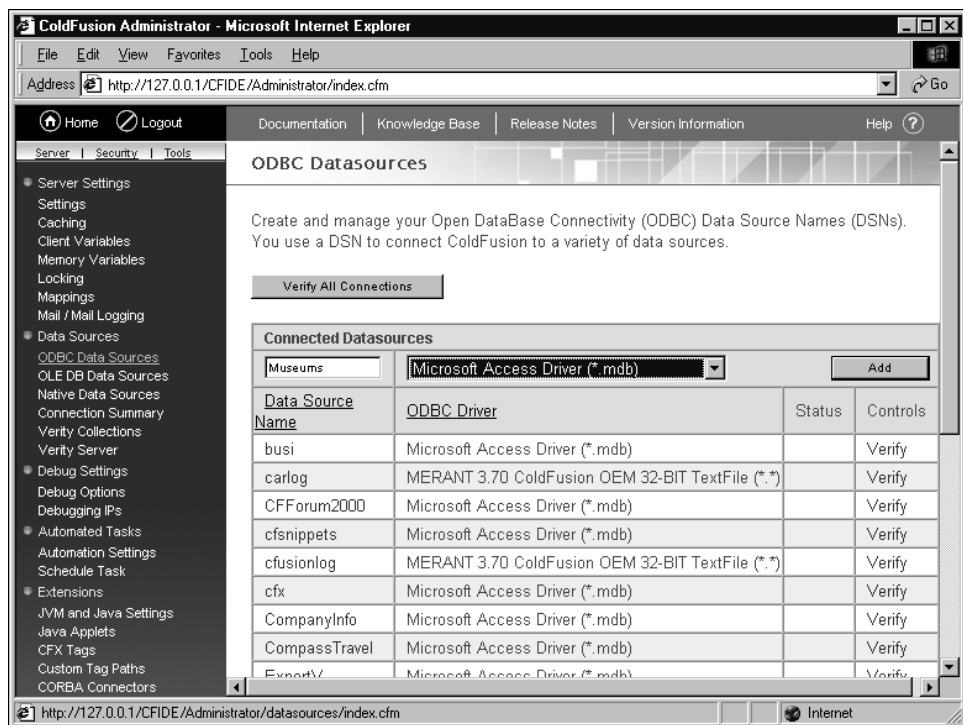


Рис. 1.8. Создание источника данных

Система администрирования ColdFusion включает средство для проверки доступности источника данных. Не забывайте, что доступ может быть огра-

ничен в случае неправильного указания имени пользователя и пароля для данного источника. Чтобы проверить сразу все источники данных, достаточно нажать кнопку **Verify All Connections** (Проверить все соединения). На рис. 1.9 приведен пример результата проверки источников данных.

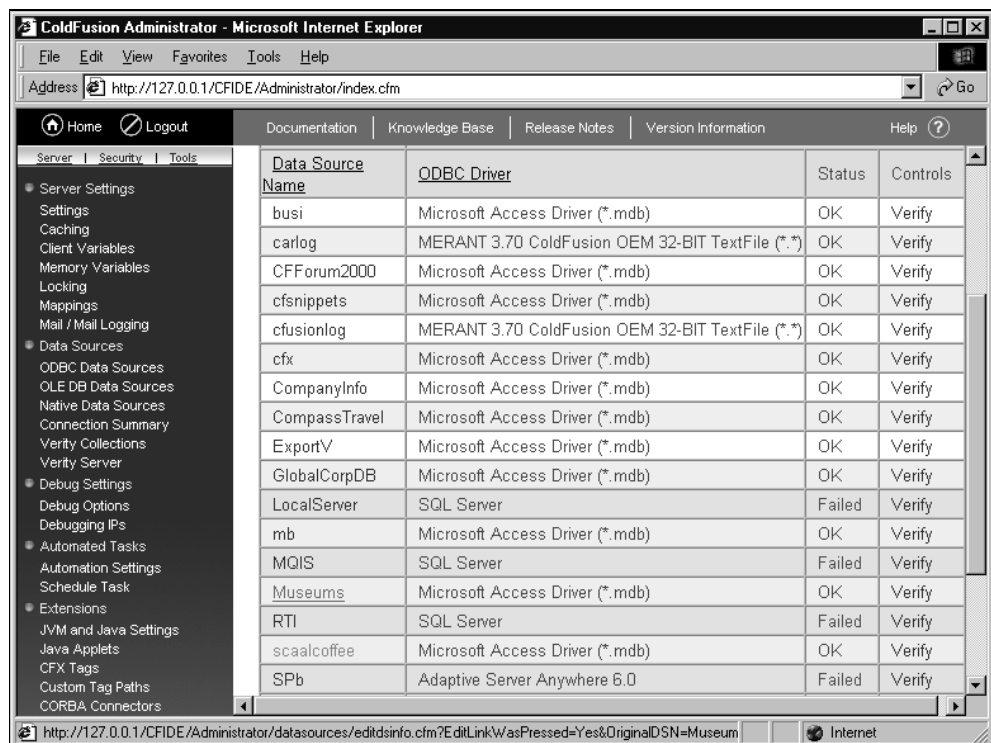


Рис. 1.9. Результат проверки доступности источников данных

В списке источников данных в столбце **Status** (Состояние), отобразится результат тестирования. Впрочем, также можно проверить индивидуальный источник данных, щелкнув напротив наименования источника по ссылке под названием **Verify** (Проверено).

В ходе постижения возможностей ColdFusion мы будем возвращаться к службе администрирования в соответствии с изучаемым материалом.

Итак, вы установили ColdFusion Server и поэкспериментировали с его настройками. В общем, это уже неплохо. На данном этапе ваш Web-браузер в состоянии будет "разобраться" со страницами ColdFusion, имеющими расширение cfm, и отразить их содержимое в соответствии с правилами языка ColdFusion Markup Language (CFML).



Установка и настройка ColdFusion Studio 5

ColdFusion Studio — это интегрированная среда для высокопроизводительной визуальной разработки Web-приложений на платформе ColdFusion Server. В ее основу положен редактор Allaire HomeSite, дополненный большим количеством визуальных инструментов для работы с базами данных, программирования и отладки, обеспечивающих удобство коллективной разработки, управления проектами и развертывания готовых серверных приложений. Продукт ColdFusion Studio тесно интегрируется с Macromedia Dreamweaver, Microsoft Office, Watchfire Linkbot.

Системные требования

Для установки и успешного использования ColdFusion Studio 5 ваша система должна выдерживать следующие минимальные требования:

- ☐ процессор, совместимый с Pentium;
- ☐ Windows 98/ME, Windows NT 4.0 или Windows 2000;
- ☐ Internet Explorer 4.5 или более поздней версии;
- ☐ 64 Мбайт оперативной памяти (рекомендуется 128 Мбайт);
- ☐ 30 Мбайт свободного пространства на жестком диске;
- ☐ устройство CD-ROM;
- ☐ экранное разрешение 800×600;
- ☐ наличие ColdFusion Server 5 (для доступа к базам данных, отладки и разработки).

Установка ColdFusion Studio

Для установки ColdFusion Studio с компакт-диска предварительно следует закрыть все запущенные приложения и выполнить следующие действия.

1. Вставьте в устройство CD-ROM установочный компакт-диск. После чего будет загружен экран ColdFusion Studio с приглашением к инсталляции программы (**Welcome**).

2. Нажмите кнопку **Next** (Далее). Будет запущен мастер установки InstallShield Wizard.
3. Нажмите кнопку **Yes** (Да), подтверждая тем самым, что вы ознакомлены и согласны с лицензионным соглашением программного продукта (**License Agreement**).
4. Введите пользовательскую информацию (**Customer Information**), имя пользователя (**User Name**), название вашей компании (**Company Name**) и серийный номер продукта (**Serial Number**), после чего нажмите кнопку **Next** (Далее).
5. Если вы хотите изменить местоположение программного продукта (**Choose Destination Path**), нажмите кнопку **Browse** (Обзор) и укажите необходимый путь, а затем нажмите кнопку **Next** (Далее).
6. Программа установки предложит выбрать необходимые для инсталляции компоненты (**Select Components**). Если вы укажете вариант установки с документацией (**Documentation**), то потребуется дополнительное пространство на жестком диске в размере 12 Мбайт. После соответствующего выбора нажмите кнопку **Next** (Далее).
7. Следующим шагом вместе с установкой ColdFusion Studio 5 предлагается установить редактор каскадных таблиц стилей — TopStyle Lite 2.1 от компании Bradbury Software. Выберите расположение программного продукта TopStyle Lite, используя кнопку **Browse** (Обзор), и нажмите кнопку **Next** (Далее).
8. Ознакомившись с информацией о выбранных выше параметрах установки, подтвердите свой выбор нажатием кнопки **Next** (Далее). Вы увидите окно, отображающее начальную стадию копирования файлов.
9. После окончания копирования файлов мастер установки предложит перезагрузить компьютер (**Restart**) для вступления в силу установленных компонентов нового программного продукта. Произведите выбор и нажмите кнопку **Finish** (Готово).

После удачной инсталляции поверх предыдущей версии при первой загрузке установленного ColdFusion Studio 5 автоматически запускается мастер предыдущей конфигурации (Initial Configuration Wizard), позволяющий импортировать настройки и другие назначения в новую версию. Однако, отказавшись от импорта настроек предыдущей версии при первой загрузке ColdFusion Studio 5, можно исправить данное упущение, воспользовавшись главным меню программы и выбрав команду **File | New** (Файл | Создать), перейдя на вкладку **Custom** (Настройка) и щелкнув по значку **Initial Configuration Wizard** (Мастер предыдущей конфигурации), как это показано на рис. 2.1.

Список пунктов, которые будут предоставлены для импортирования в новую версию, таков:

- ☐ **Remote Servers** (Удаленные серверы);
- ☐ **Tools Bars** (Панели инструментов);

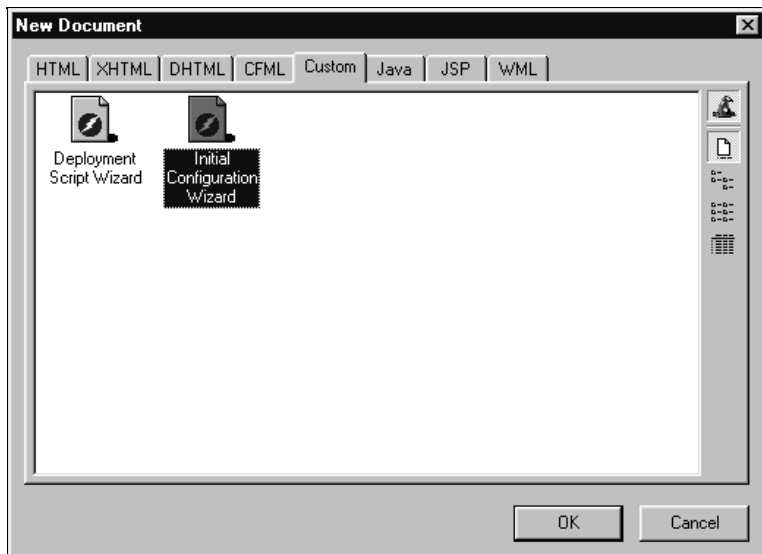


Рис. 2.1. Диалоговое окно **New Document**, вкладка **Custom**

- ☐ **Feature Customization** (Пользовательские особенности) — фрагменты кодов для многочисленного использования (**Snippets and Snippet Shortcuts**), шаблоны кодов (**Code Templates**), автоматическая простановка тегов (**Tag Insight**) и т. д.;
- ☐ **Editor Settings** (Параметры редактора);
- ☐ **General Setting** (Общие параметры);
- ☐ **Miscellaneous Setting** (Различные установки) — цветовая палитра (**Color Palettes**), избранные каталоги (**Favorite Folders**), автоматизированный процесс форматирования кода (**Code Sweeper**) и т. д.;
- ☐ **Debug Settings** (Параметры отладки);
- ☐ **Debug Mappings** (Логические псевдонимы для отладки страниц);
- ☐ **Development Styles** (Стили разработки).

Настройки ColdFusion Studio

Следует отметить, что в данном редакторе модифицируется практически все, от цветов раскраски тегов и клавиатурных комбинаций быстрого доступа до алгоритмов проверки правильности написания страниц и способов очистки кодов, что позволяет настроить программу в соответствии с вашими предпочтениями и вкусами.

Вы можете настроить ColdFusion Studio на свой лад, выбрав команду **Options | Settings** (Настройка | Параметры) или нажав клавишу <F8>. На рис. 2.2

показано окно настроек ColdFusion Studio. В левой части окна расположен иерархический список, позволяющий разделить большое количество настраиваемых элементов по принадлежности. Это достаточно удобно. При щелчке по одному из пунктов данного списка в правой части окна настроек появляется набор элементов, относящихся к данному разделу.

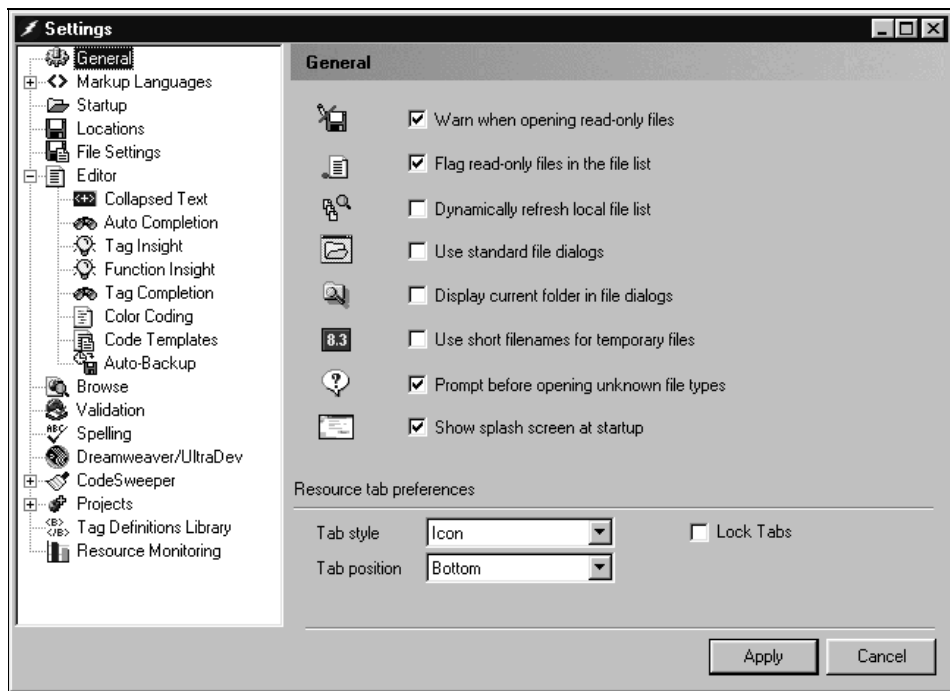


Рис. 2.2. Диалоговое окно **Settings** в ColdFusion Studio

Автор книги не преследует цели описать буквально все настраиваемые элементы программного продукта ColdFusion Studio, тем более что многие из них интуитивно понятны. Однако кое о чем следует рассказать.

Перекодирование документов

Кодировка представляет собой таблицу символов, где каждой букве алфавита (а также цифрам и специальным знакам) присвоен свой уникальный номер — *код символа*. В настоящий момент существуют различные кодировки, используемые для хранения Web-документов. ColdFusion Studio позволяет сохранять Web-страницы в следующих кодировках:

☐ ANSI;	☐ Unicode;
☐ Unicode Big Endian;	☐ UTF-8.

Для поддержки сохранения документов в одной из перечисленных кодировок следует в диалоговом окне настроек **Settings** (Параметры), выбрать раздел **File Settings** (Параметры файлов) и отметить флажок **Enable not-ANSI file encoding** (Использовать не-ANSI кодировку файлов), как показано на рис. 2.3.

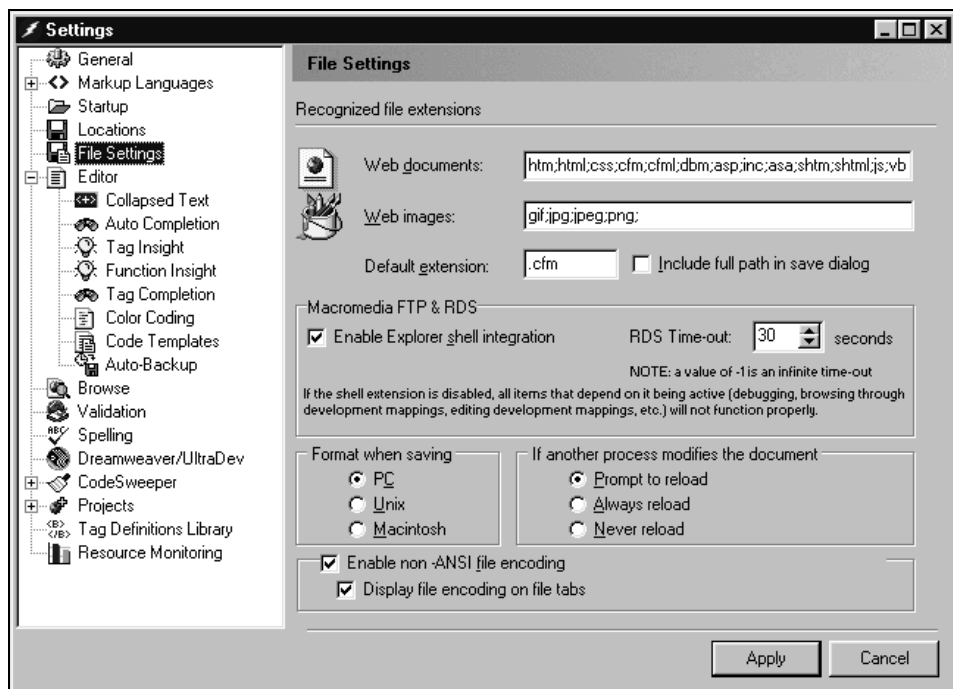


Рис. 2.3. Диалоговое окно **Settings**, раздел **File Settings**

Затем для перекодирования текущего документа следует выбрать команду **File | Save As** (Файл | Сохранить как) или нажать комбинацию клавиш **<SHIFT>+<CTRL>+<S>** и в диалоговом окне сохранения документа указать одно из предлагаемых значений кодировки в раскрывающемся списке **Encoding** (Кодировка).

Флажок **Display file encoding on file tabs** (Отображать файл в разных кодировках), расположенный все в том же разделе файловых настроек **File Settings** (Параметры файлов), позволяет выводить каждую страницу, открытую в рабочей области ColdFusion Studio, в определенной кодировке, как отражено на рис. 2.4.

Также в разделе файловых настроек можно:

- указать поддерживаемые расширения как для Web-документов (поле **Web documents**), так и для графических фалов, используемых в Web-приложениях (поле **Web images**);

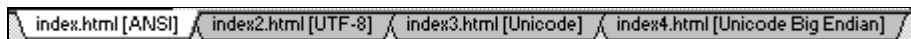


Рис. 2.4. Отражение кодировок открытых документов в среде ColdFusion Studio

- ☐ установить расширение по умолчанию (поле **Default extension**);
- ☐ включить поддержку FTP- и RDS-серверов (группа **Macromedia FTP & RDS**); этот процесс подробно описан в разд. "Установка соединения с удаленным сервером" далее в этой главе;
- ☐ настроить формат сохраняемых документов (группа **Format when saving**), который может принимать такие значения как **PC**, **Unix** или **Macintosh**.

Шрифты и сворачиваемый текст

Если при написании страниц вы используете язык, не принадлежащий к группе латинских кодировок, то скорее всего вы можете столкнуться с проблемой отображения текстов в режиме редактирования.

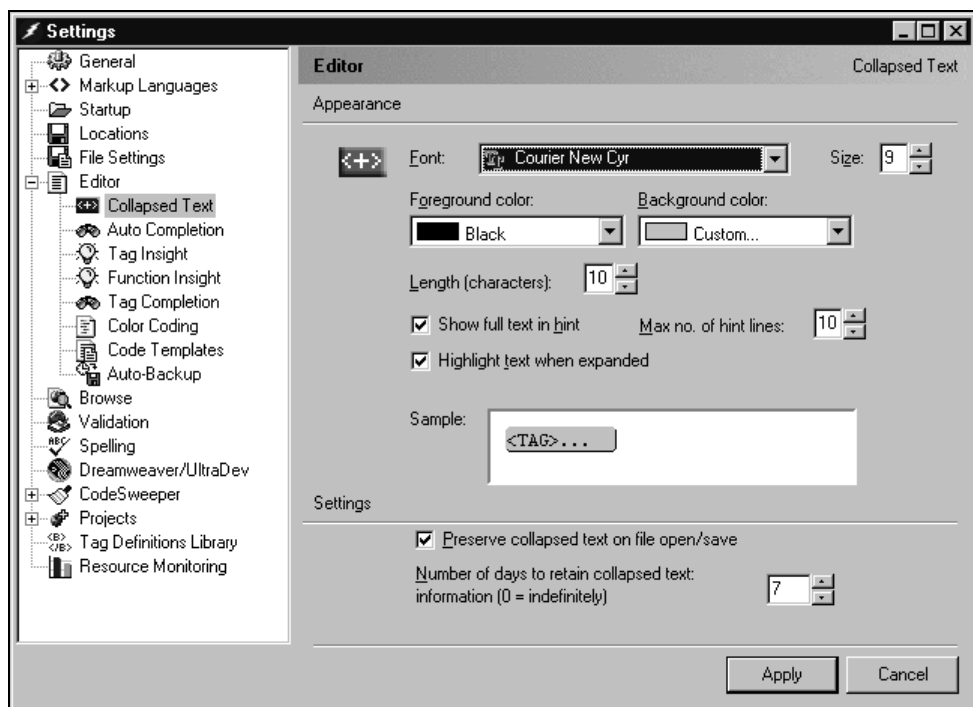


Рис. 2.5. Диалоговое окно **Settings**, раздел **Editor | Collapsed Text**

Например, для правильного отображения кириллического текста достаточно выбрать команду **Options | Settings** (Настройка | Параметры), перейти к раз-

делу **Editor** (Редактор) и в раскрывающемся списке **Font** (Шрифт) указать кириллический шрифт. Также в разделе **Editor**, используя список **Charset** (Набор), можно выбрать значение *Russian*. Более того, если вам придется пользоваться достаточно удобным средством сворачивания текста (**Collapsed Text**, очень похоже на средство скрытия фрагментов в MS Excel), то следует перейти к подразделу **Collapsed Text** (Сворачиваемый текст) и в раскрывающемся списке **Font** (Шрифт) задать кириллический шрифт (рис. 2.5).

Здесь же вы можете указать цвет и размер шрифта, цвет фона, определить, демонстрировать ли текст при перемещении курсора мыши над фрагментом свернутого текста и в каком объеме (по умолчанию 10 строк текста). Не выходя из диалога, в поле **Sample** (Образец) можно увидеть пример, демонстрирующий, как будет выглядеть фрагмент свернутого текста.

Автоматическая вставка тегов

Если вам доводилось программировать на Visual Basic или использовать Visual Basic for Applications в одном из офисных продуктов Microsoft, то вы наверняка сталкивались с очень удобным средством подсказок, когда, заканчивая ввод того или иного объекта точкой, вы получали список относящихся к этому объекту свойств, или завершая ввод очередной функции круглой скобкой, замечали подсказку в виде перечня доступных этой функции параметров. В ColdFusion Studio встроено похожее средство — автоматическая вставка тегов (**Tag Insight**).

На рис. 2.6 показано диалоговое окно настроек **Tag Insight** (Список тегов).

В этом разделе можно включать и отключать данный сервис, указывать время реакции в секундах, добавлять собственные теги и удалять существующие. И если этот сервис активен, то достаточно в режиме редактирования текста ввести символ открытия тега (<) и задержаться на мгновение (в нашем случае на одну секунду), нашему взору откроется список подсказки с перечнем существующих в разделе **Tag Insight** (Список тегов) тегов (рис. 2.7).

Выбрав необходимый тег из предоставленного выше списка, нажмите клавишу <Пробел> и опять же задержитесь на мгновение. Вы увидите еще один список подсказок, уже с набором возможных атрибутов для выбранного тега. При выборе тегов или атрибутов помните, что кроме вертикальной полосы прокрутки, находящейся с правой стороны списка, можно пользоваться вводом начальных символов, что порой облегчает поиск тегов и их атрибутов.

Аналогично можно включать и отключать средство подсказки функций ColdFusion, используя раздел **Editor | Function Insight** (Редактор | Список функций) все в том же диалоговом окне настроек.

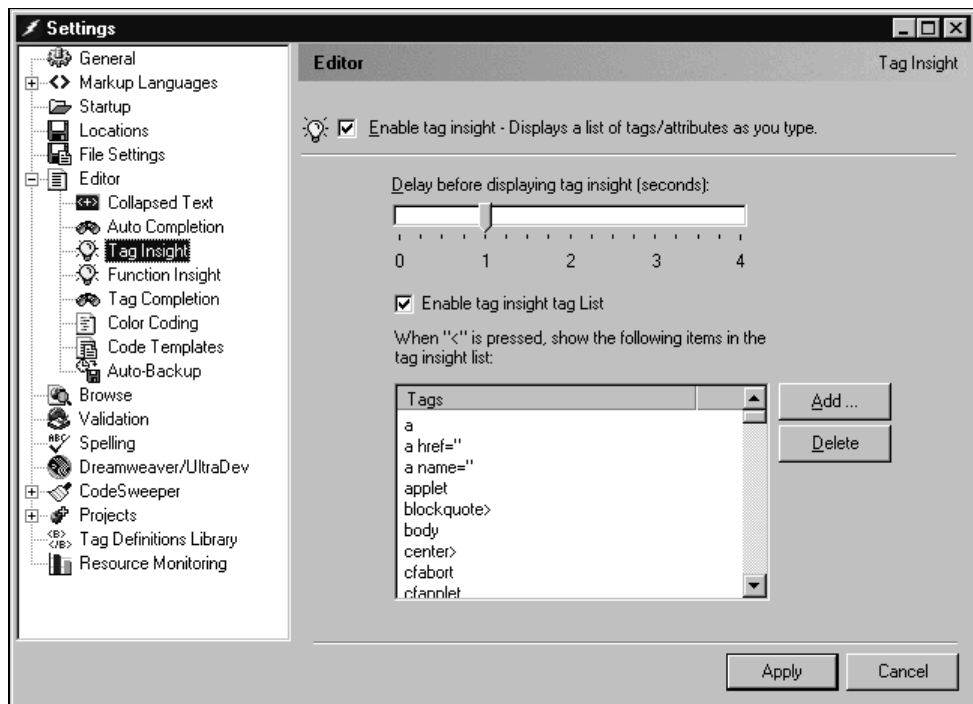


Рис. 2.6. Диалоговое окно **Settings**, раздел **Editor | Tag Insight**

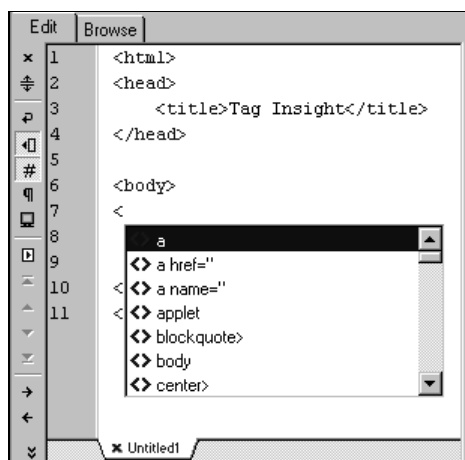


Рис. 2.7. Перечень существующих тегов

Цветовая схема элементов кодирования

Редактор ColdFusion Studio имеет возможность комбинировать цвета различных элементов кодирования. Для настройки цветовых схем используйте раздел **Editor | Color Coding** (Редактор | Цветовое кодирование) в диалоговом окне настроек (рис. 2.8).

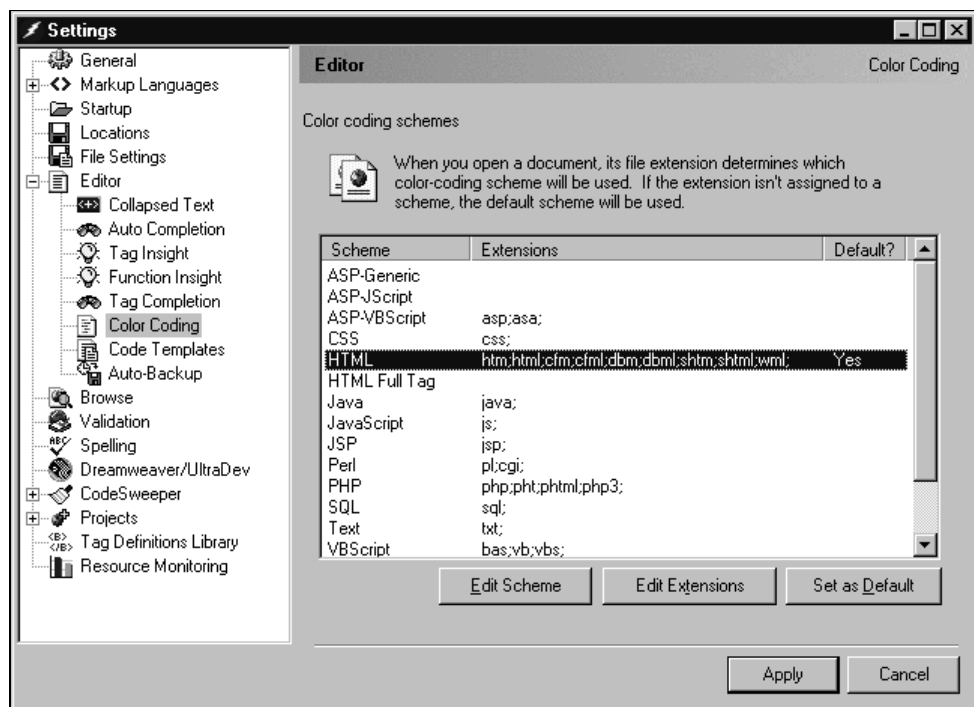


Рис. 2.8. Диалоговое окно **Settings**, раздел **Editor | Color Coding**

Для редактирования схемы достаточно выбрать необходимую цветовую схему и нажать кнопку **Edit Scheme** (Правка схемы). В результате откроется дополнительное диалоговое окно, разделенное на три области. В левой верхней части расположен список элементов, в правой верхней части — набор опций для редактирования параметров текущего элемента, а в нижней части окна отображается пример кода в контексте данных настроек (рис. 2.9).

Для выбора текущего элемента можно пользоваться как списком элементов, так и щелкая по элементам в приведенном примере.

Обратите еще раз ваше внимание на рис. 2.8. В этом же разделе имеется возможность изменять связанные расширения файлов по отношению к той или иной схеме. Для этого достаточно нажать кнопку **Edit Extensions** (Правка расширений), где все расширения перечислены через точку с запя-

той. А с помощью кнопки **Set as Default** (По умолчанию) можно указать схему, используемую редактором по умолчанию.

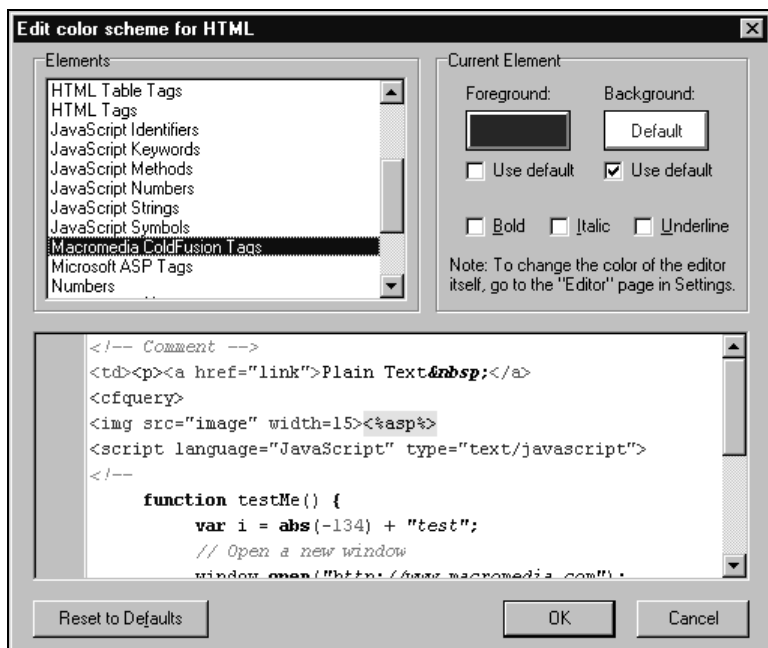


Рис. 2.9. Диалоговое окно настроек редактирования цветовой схемы

В качестве заключения по цветовым схемам ваших замечательных кодов хотелось бы отметить, что цвета, подобранные разработчиками ColdFusion Studio, достаточно удачны, так что прежде чем что-либо менять, попробуйте создать свою первую страницу без изменения цветовых схем.

Шаблоны кодов

Нельзя не упомянуть о шаблонах кодов (**Code Templates**). Данный сервис позволяет воспользоваться как уже существующими шаблонами, так и добавить собственные. Для настройки редактирования или добавления новых шаблонов существует раздел **Editor | Code Templates** (Редактор | Шаблоны кода) в диалоговом окне настроек (рис. 2.10).

Шаблон подразумевает некий код, имеющий собственное ключевое слово и описание. Для вставки шаблонов в режиме редактирования необходимо нажать комбинацию клавиш <Ctrl>+<J>.

Мы еще вернемся к теме использования собственных шаблонов (см. разд. "Множественно повторяющиеся фрагменты кода" гл. 4).

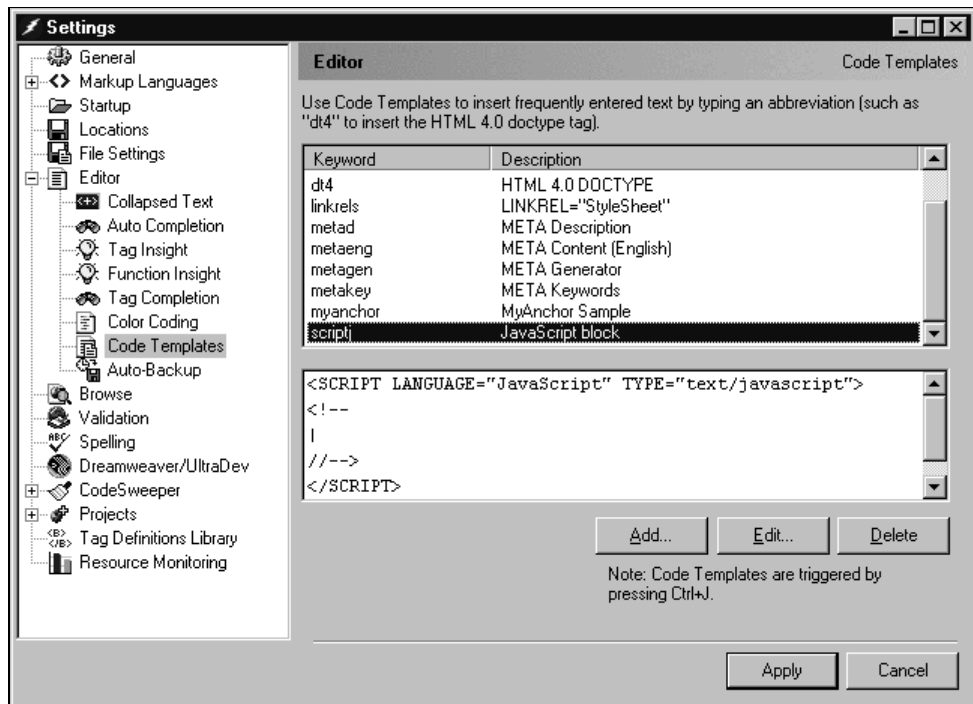


Рис. 2.10. Диалоговое окно **Settings**, раздел **Editor | Code Templates**

Автоматическое резервирование

В ColdFusion Studio встроена служба автоматического резервирования документов. Если у вас возникает необходимость восстанавливать содержание утраченных по какой-либо причине документов, то службу автоматического резервирования (**Auto-Backup**) следует держать включенной. На рис. 2.11 продемонстрирован раздел **Editor | Auto-Backup** (Редактор | Автоматическое резервирование).

В этом же разделе вы можете регулировать месторасположение резервируемых документов (**Backup Directory**), способ сохранения резервируемых документов, количество дней хранения резервируемых документов перед удалением и интервал резервирования документов (**Timed Auto-Backup**), по умолчанию равный 10 минутам.

Вопросы по восстановлению документов будут рассмотрены в разд. "Восстановление резервных копий" гл. 4.

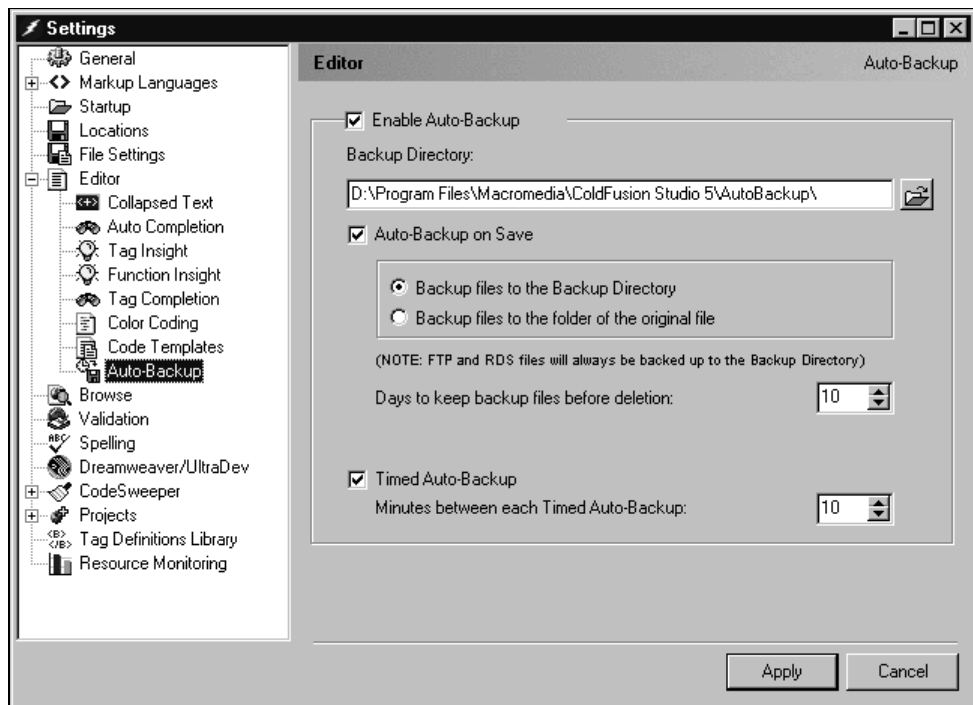


Рис. 2.11. Диалоговое окно **Settings**, раздел **Editor | Auto-Backup**

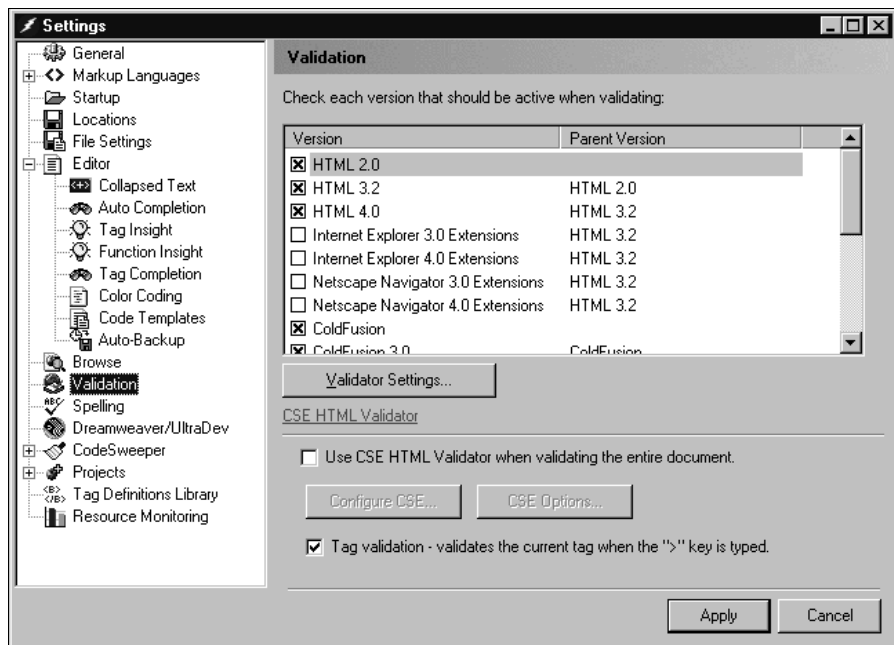
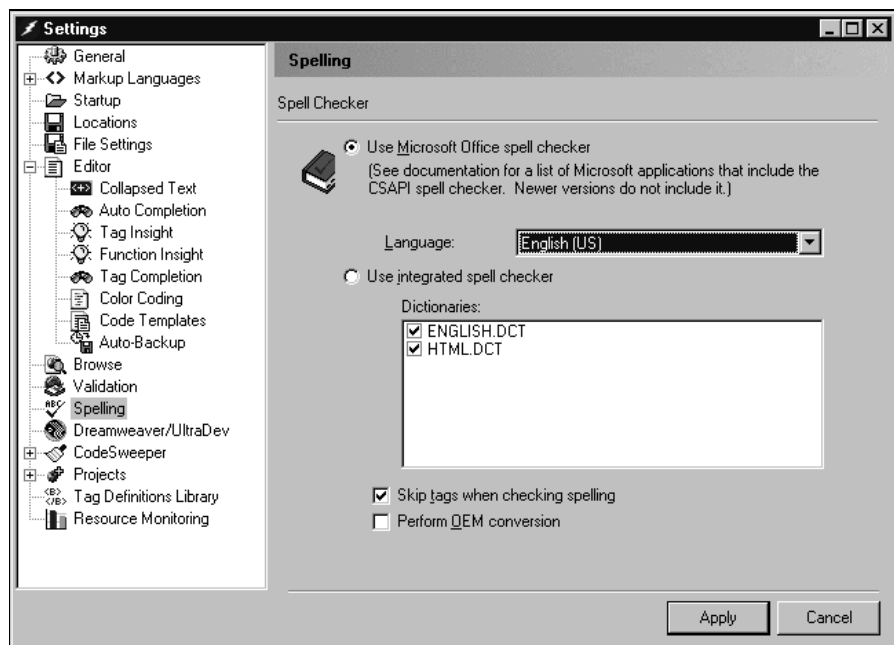
Проверка правильности написания страниц

В ColdFusion Studio входит сервис проверки правильности написания страниц. Для настройки данного сервиса используйте раздел **Validation** (Допустимость) в диалоговом окне настроек (рис. 2.12).

В этом разделе находится список поддерживаемых спецификаций, напротив каждой из которых расположен флажок. Его состояние задает, проверять страницу по данной спецификации или нет. Для указания дополнительных настроек нажмите кнопку **Validator Settings** (Параметры проверки), где вы сможете более подробно указать, на что следует обращать внимание при анализе страницы и что необходимо включить в отчет результата проверки страниц.

Правописание

ColdFusion Studio имеет встроенные функции проверки орфографии английского языка, а также позволяет работать со словарями Microsoft Office 95/97/2000, что достаточно удобно для разработки неанглоязычных страниц. Чтобы настроить проверку орфографии, используйте раздел **Spelling** (Орфография) в диалоговом окне настроек (рис. 2.13).

Рис. 2.12. Диалоговое окно **Settings**, раздел **Validation**Рис. 2.13. Диалоговое окно **Settings**, раздел **Spelling**

В этом разделе имеются флажки, первый из которых позволяет при проверке орфографии пропускать теги, второй — выполнять преобразование в OEM-кодировку. Чтобы начать процесс проверки орфографии, достаточно из главного меню приложения либо выбрать команду **Tools | Spell Check** (Сервис | Проверка орфографии) или нажать клавишу <F7> (указанная команда дублируется на панели инструментов **Tools** (Сервис)), либо указать команду **Tools | Spell Check All** (Сервис | Проверить орфографию всего) или нажать комбинацию клавиш <Shift>+<F7>, в зависимости от необходимости. Более того, можно воспользоваться командой **Tools | Mark Spelling Errors** (Сервис | Проверять орфографию в режиме редактирования) или нажать комбинацию клавиш <Ctrl>+<F7>, что позволяет непосредственно в режиме редактирования выделять все слова, в которых были допущены ошибки. Слова выделяются подчеркнутой волнистой линией красного цвета. Данная команда также дублируется на панели инструментов **Tools** (Сервис).

Автоматизация процесса формирования кода

Функция автоматизированного процесса форматирования кода (**CodeSweeper**) может быть использована в ряде случаев:

- ☐ после визуального редактирования элементов страницы в режиме **Design view** (Режим дизайна). Вы можете настроить **CodeSweeper** (Автоформат) так, чтобы форматирование кода производилось при выходе из режима **Design** (Дизайн) — для более ранних версий;
- ☐ для "подчистки" кода при просмотре документов существующих проектов;
- ☐ для приведения стиля написания кода группой разработчиков к единой концепции форматирования.

Редактор ColdFusion Studio включает в себя несколько профилей автоматизации форматирования кода. Вы можете также создавать свои собственные профили **CodeSweeper** или изменять уже существующие. В ColdFusion Studio имеются следующие варианты автоформатирования **CodeSweeper**:

- ☐ **JSP CodeSweeper** — формирует код в стиле Java Server Pages;
- ☐ **Allaire Default CodeSweeper (Optimized for HTML Only)** (По умолчанию) — оптимально подходит для стандартных Web-страниц;
- ☐ **Allaire Default CodeSweeper (HTML\CFML)** — форматирует прикладные программы GoldFusion;
- ☐ **My CodeSweeper** — копия **CodeSweeper**, которую можно использовать для тестирования;
- ☐ **WDDX Sweeper** — копия **CodeSweeper** форматирует код в стиле Web Distributed Data Exchange;
- ☐ **WEB-XML CodeSweeper** — формирует код в стиле Extensible Markup Language;

- ☐ **Macromedia Default HTML Tidy Settings** — формирует код в стиле Macromedia;
- ☐ **Macromedia HTML Tidy Settings** — формирует код в стиле Macromedia, практически дублирует **Macromedia Default HTML Tidy Settings** за исключением пары значений.

Для настройки автоматизации процесса формирования кода выберите в диалоговом окне настроек раздел **CodeSweeper** (Автоформат) — рис. 2.14.

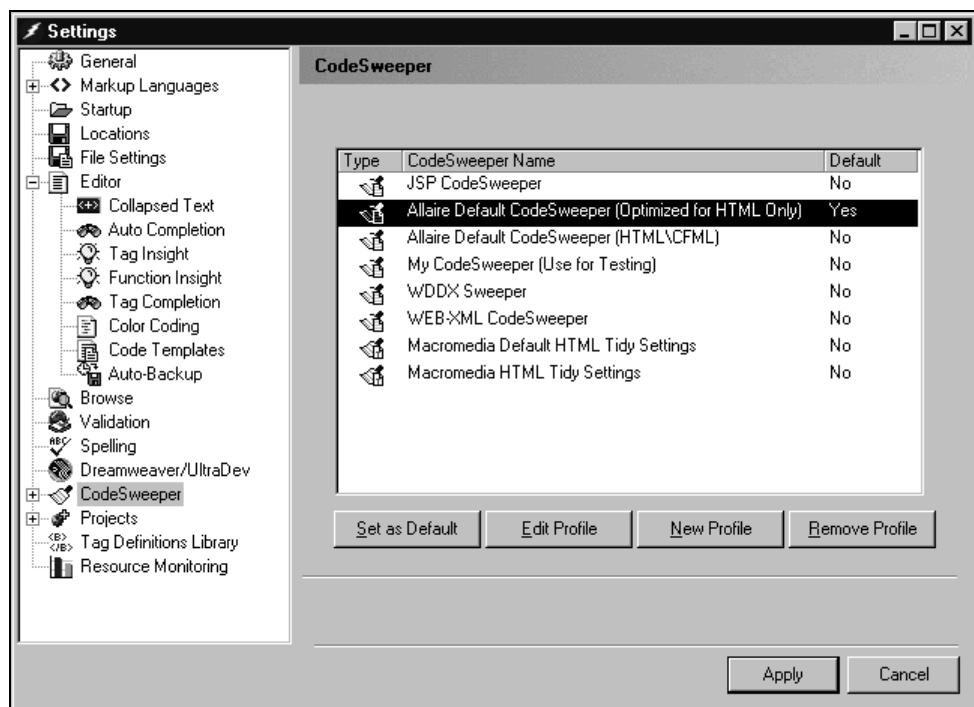


Рис. 2.14. Диалоговое окно **Settings**, раздел **CodeSweeper**

Обратите внимание, что в этом разделе имеется возможность устанавливать профиль формирования кода по умолчанию, используя кнопку **Set as Default** (По умолчанию). Для редактирования профиля служит кнопка **Edit Profile** (Правка профиля). При создании нового профиля необходимо:

1. Нажать кнопку **New Profile** (Новый профиль).
2. Заполнить диалоговое окно **New CodeSweeper Profile** (Новый профиль автоформата) — рис. 2.15.
3. Ввести имя **CodeSweeper** (Автоформат) и выбрать тип: **Macromedia CodeSweeper** или **HTML Tidy CodeSweeper**.

4. Приступить к редактированию созданного профиля в зависимости от ваших потребностей.

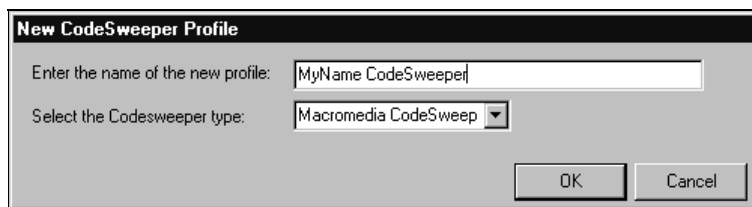


Рис. 2.15. Диалоговое окно создания нового профиля
New CodeSweeper Profile

Для удаления профиля достаточно, выделив его из предлагаемого списка в разделе **CodeSweeper** (Автоформат) диалогового окна настроек среды ColdFusion Studio, нажать кнопку **Remove Profile** (Удалить профиль).

Существуют следующие виды настроек для **CodeSweeper** (Автоформат):

- ☐ общие настройки (**General Settings**) и специальные настройки (**Tag-specific Settings**) могут быть установлены для каждого варианта **Macromedia CodeSweeper**;
- ☐ настройки **HTML Tidy** относятся только к типу **HTML Tidy CodeSweeper**.

Настройки **General Settings**

Вы можете задать приведенные ниже правила форматирования для каждого типа **Macromedia CodeSweeper**.

- ☐ Устанавливать регистр букв для тегов и наименований атрибутов. Существует возможность выбрать как верхний (**All Upper Case**) или нижний регистр (**All Lower Case**), так и сохранить регистр без изменений (**Preserve Case**), используя раскрывающиеся списки **Format tag names** (Форматировать имена тегов) и **Format attribute names** (Форматировать имена атрибутов).
- ☐ Настроить формат имен событий **Format event names** (Форматировать имена событий), включая дополнительную опцию, а именно смешанный регистр (**Mixed Case**), который применим для имен событий JavaScript, таких как `OnMouseOver`.
- ☐ Установить кавычки для значений атрибутов с помощью раскрывающегося списка **Attribute value quoting** (Кавычки для значений атрибутов).
- ☐ Удалить пробелы между тегами с помощью установленного по умолчанию флажка **Trim white space between tags** (Убрать пробел между тегами). Рекомендуется оставлять данную настройку как есть, отключая только при необходимости для отдельных тегов.

- ☐ Установить режим работы "silent mode" во избежание предупреждений, которые может генерировать средство **CodeSweeper** (Автоформат) с помощью флажка **Run in "silent mode"** (Запускать в режиме "silent mode"). Вы можете использовать эту опцию для работы **CodeSweeper** (Автоформат) в автоматическом режиме.
- ☐ Установить протоколирование ошибок в log-файл, отметив флажок **Write errors to a log file** (Записывать ошибки в журнал) с целью обнаружения каких-либо ошибок, найденных **CodeSweeper** (Автоформат). Вы можете сами определить место протокола (или, как его еще называют, системного журнала).

Не забудьте нажать кнопку **ОК** для сохранения настроек (рис. 2.16).

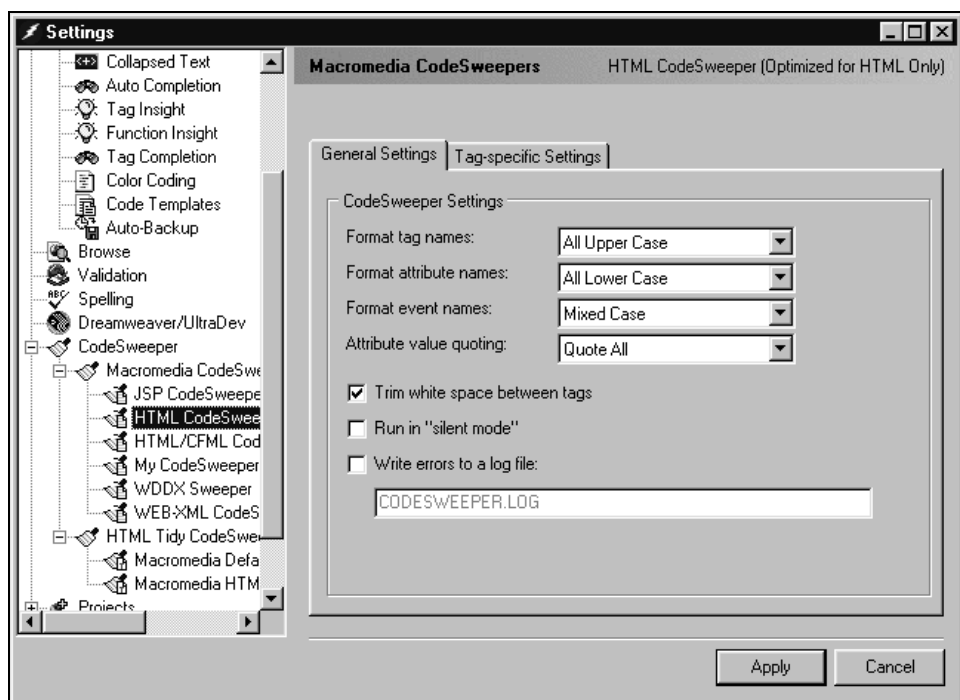


Рис. 2.16. Диалоговое окно **Settings**, раздел **Macromedia CodeSweepers | HTML CodeSweeper (Optimized for HTML Only)**, вкладка **HTML General Settings**

Настройки *Tag-specific Settings*

Широкий выбор опций существует для индивидуальных тегов, содержащихся в каждом варианте автоформата **Macromedia CodeSweeper**. Если вы устанавливаете тип форматирования **All Other Tags** (Все другие теги), то это правило будет действовать в отношении каждого тега, появляющегося в доку-

менте и не указанного в списке диалогового окна **Settings** (Параметры) — рис. 2.17. Здесь вы можете:

- ☐ вставлять команды с новой строки для первого и последнего тегов;
- ☐ устанавливать отступы с помощью табуляции или пробелов — разделитель строк добавляется автоматически для каждой новой строки;
- ☐ разрешать включать разделители строк для вложенных тегов (sub tags);
- ☐ вручную управлять установками пробелов путем предотвращения "обрезания" пробелов вокруг выделенного тега;
- ☐ оставлять выбранный тег неизменным при активизации средства **CodeSweeper** (Автоформат);
- ☐ изолировать тег в документе — это удобно для избавления от ненужных тегов, вставленных генератором кода.

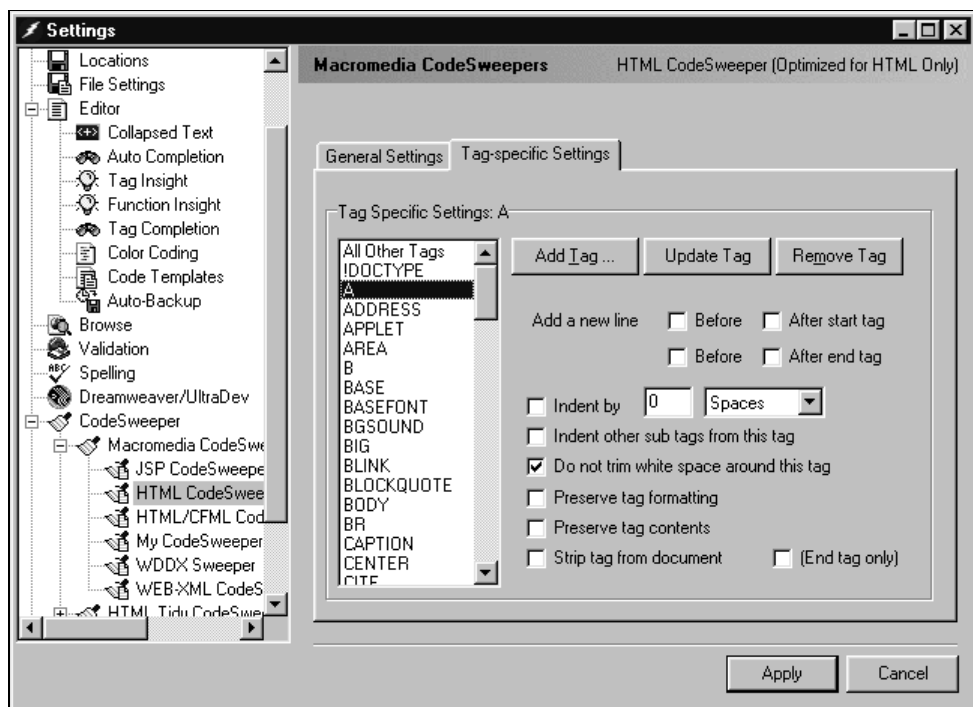


Рис. 2.17. Диалоговое окно **Settings**, раздел **Macromedia CodeSweepers | HTML CodeSweeper (Optimized for HTML Only)**, вкладка **Tag-specific Settings**

Установка правил для тега может быть произведена в следующем порядке:

1. Выберите тег из списка **Tag Specific Settings** (Параметры особого тега) в диалоговом окне редактирования настроек раздела **CodeSweeper** (Автоформат).

2. Измените специальные настройки.
3. Нажмите кнопку **Update Tag** (Обновить тег) для сохранения настроек данного тега.

Добавление тега осуществляется следующим образом:

1. Нажмите кнопку **Add Tag** (Добавить тег) в диалоговом окне редактирования настроек **CodeSweeper** (Автоформат).
2. Введите имя тега и нажмите на кнопку **OK** для подтверждения своих действий.
3. Произведите изменения настройки тега и нажмите кнопку **Update Tag** (Обновить тег) для сохранения настроек.

Удаление тега:

1. Выберите тег из списка **Tag Specific Settings** (Параметры особого тега) в диалоговом окне редактирования настроек **CodeSweeper** (Автоформат).
2. Нажмите кнопку **Remove Tag** (Удалить тег).
3. Закройте диалоговое окно нажатием кнопки **OK**.

Настройки *HTML Tidy CodeSweeper*

Для настроек **HTML Tidy CodeSweeper** вы можете установить перечисленные ниже правила форматирования.

- ☐ Необходимо ли указывать форматируемые опции из существующего списка.
- ☐ Следует ли устанавливать кодировку и тип документа.
- ☐ Определить максимальную ширину страницы, величину пробелов и табуляций.
- ☐ Нужно ли устанавливать режим работы "silent mode" во избежание предупреждений, которые может генерировать средство **CodeSweeper** (Автоформат).
- ☐ Указать название log-файла (файл системного журнала) для протоколирования выполнения процесса формирования кода.

Вы можете сформировать собственный вариант форматирования **HTML Tidy CodeSweeper**, создав файл с расширением `tdy` и сохранив его в каталоге ColdFusion Studio — `\Extensions\CodeSweepers`.

Чтобы выполнить процесс автоматизированного форматирования кода, необходимо из главного меню приложения выбрать команду **Tools | CodeSweeper** (Сервис | Автоформат), данная команда дублируется на панели инструментов **Edit** (Правка).

Важно подчеркнуть, что после выполнения автоматизированного форматирования кода команда **Undo** (Отменить), позволяющая отказаться от последних совершенных действий по редактированию документа, недоступна.

В дополнение

Следует отметить, что в разделе **Dreamweaver/UltraDev** диалогового окна настроек **Settings** (Параметры) находится переключатель, позволяющий интегрировать ColdFusion Studio с редактором Macromedia Dreamweaver. На взгляд автора, это достаточно удачная связка, поскольку одно формирование страниц удобно выполнять в ColdFusion Studio, а другое — в Dreamweaver. Впрочем, кому как нравится. Здесь сила привычки играет не последнюю роль.

Редактор ColdFusion Studio предлагает удобную возможность предварительного просмотра страниц. Данный режим позволяет увидеть, как "на деле" будет выглядеть результат программирования. Функция предварительного просмотра предоставляет возможность выбора просмотра как в Microsoft Internet Explorer, так и в Netscape Navigator. Для задания браузера необходимо выбрать команду **Options | Settings | Browse** (Настройка | Параметры | Просмотр).

В разд. "Просмотр CFML-страниц" гл. 6 более детально остановимся на данной конфигурации.

Установка соединения с удаленным сервером

Так или иначе, при создании того или иного Web-сайта сталкиваешься с необходимостью публикации этого сайта в глобальной сети Internet. И здесь на помощь приходят средства, позволяющие связываться с удаленными компьютерами (серверами) для копирования файлов и каталогов с компьютера на компьютер как в одну, так и в другую сторону. Подобные программы могут использовать различные протоколы, чаще всего — FTP (File Transfer Protocol, протокол передачи файлов).

В ColdFusion Studio встроены RDS- и FTP-клиенты, по своим возможностям ничем не уступающие специализированным программам. Для настройки одного из клиентов необходимо:

1. Открыть папку **My Computer** (Мой компьютер).
2. Щелкнуть правой кнопкой мыши на значке **Macromedia FTP & RDS** (он появляется после инсталляции ColdFusion Studio).
3. Из контекстного меню выбрать пункт **Add RDS Server** (Добавить RDS-сервер) или **Add FTP Server** (Добавить FTP-сервер) — в зависимости от потребностей.
4. В появившемся окне ввести все необходимые значения параметров.

FTP-сервер

Если вы добавляете поддержку FTP-сервера (File Transfer Protocol Server) при помощи **Macromedia FTP & RDS**, после выбора пункта **Add FTP Server** (Добавить FTP-сервер) вам нужно будет ввести информацию, чтобы подготовить сервер для копирования проектов.

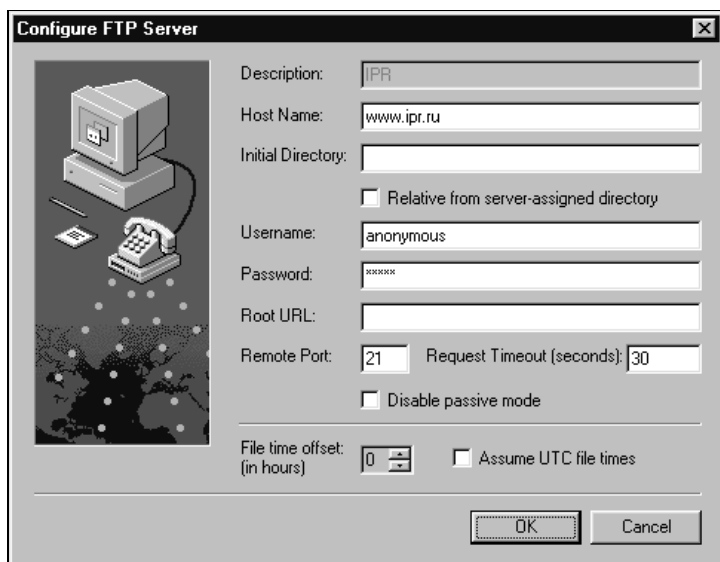


Рис. 2.18. Диалоговое окно **Configure FTP Server** для конфигурации FTP-сервера

В появившемся диалоговом окне (рис. 2.18) введите информацию для сервера, к которому хотите подключиться:

- ☐ поле **Description** (Описание) — имя, под которым будет зарегистрировано соединение с FTP-сервером;
- ☐ поле **Host Name** (Имя хоста) — имя домена, например **www.ipr.ru**, или IP-адрес. В случае, если ftp составляет часть имени домена, то необходимо ввести полное имя, например **ftp.ipr.ru**;
- ☐ поле **Initial Directory** (Первичный каталог) — этот параметр является необязательным, если по умолчанию установлена корневая папка FTP-сервера;
- ☐ флажок **Relative form server-assigned directory** (Относительно корневой папки сервера) — указывает, должен ли первичный каталог устанавливаться из пользовательской папки или корневой папки на сервере;
- ☐ поле **Username** (Имя пользователя) — имя пользователя для доступа к удаленному серверу или значение **anonymous** для безымянных FTP-

серверов. Если в данном поле значение не будет введено, то программа потребует указать имя пользователя при соединении с FTP-сервером;

- ☐ поле **Password** (Пароль) — пароль для доступа к удаленному серверу. Если поле останется пустым, то программа потребует указать пароль при соединении с FTP-сервером;
- ☐ поле **Root URL** (Корневой URL) — введите http-адрес сайта. Эта настройка позволит вам просмотреть список файлов, открытых с удаленного сервера, измененных и сохраненных;
- ☐ поле **Remote Port** (Удаленный порт) — порт, используемый FTP-сервером. Указывайте по умолчанию значение 21, если не определен другой порт администратором сервера или Internet-провайдером;
- ☐ поле **Request Timeout** (Время ожидания) — установите время ожидания соединения с сервером;
- ☐ флажок **Disable passive mode** (Отключить режим пассивного соединения) — следует установить, если сервер не использует пассивное соединение;
- ☐ флажок **Assume UTC file times** (Применить универсальную координацию времени файлов) — задайте данный параметр, если дата/время в файлах FTP-сервера установлены неверно. Применяется для серверов, использующих формат времени Universal Coordinated Time.

Для подтверждения нажмите кнопку **OK** в текущем диалоговом окне.

RDS-сервер

Если вы добавляете поддержку RDS-сервера (Remote Development Service Server) при помощи **Macromedia FTP & RDS**, после выбора пункта **Add RDS Server** (Добавить RDS-сервер) вам нужно будет уточнить параметры, чтобы подготовить сервер для копирования проектов.

В появившемся диалоговом окне (рис. 2.19) введите информацию для сервера, к которому хотите подключиться:

- ☐ поле **Description** (Описание) — имя, которое будет находиться в списке серверов **Server List** (Список серверов);
- ☐ поле **Host name** (Имя хоста) — имя домена, например **ipr.ru**, или IP-адрес;
- ☐ поле **Port** (Порт) — порт Web-сервера. Указывайте по умолчанию значение 80, если администратором сервера не определен другой порт;
- ☐ флажок **Use Secure Sockets Layer (SSL)** (Использовать SSL) — позволяет закодировать через SSL обмен между ColdFusion Studio и сервером;
- ☐ поле **User Name** (Имя пользователя) — требуется, если сервер защищен паролем;

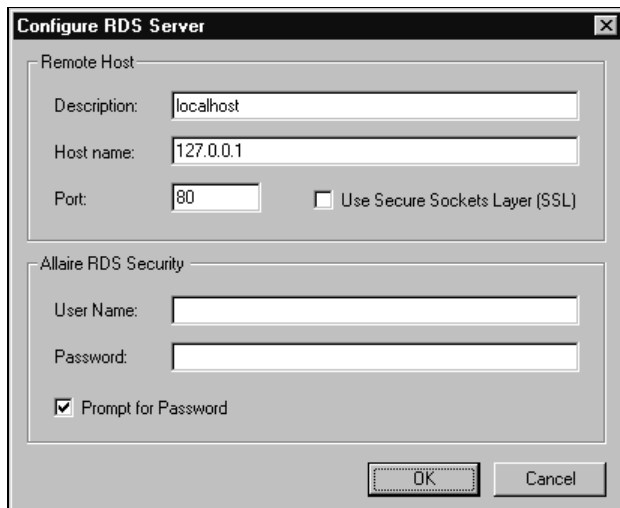


Рис. 2.19. Диалоговое окно **Configure RDS Server** для конфигурации RDS-сервера

- ☐ поле **Password** (Пароль) — требуется, если сервер защищен паролем. Для отключения защиты паролем необходимо на странице администратора "ColdFusion Administrator" в разделе **Security | Basic Security | CF Admin Password** (Защита | Основная защита | Пароль администратора CF) отключить флажок **Use a ColdFusion Administration Password** (Использовать пароль администратора ColdFusion);
- ☐ флажок **Prompt for Password** (Сообщение для пароля) — отметьте для приглашения к вводу пароля.

Для подтверждения нажмите кнопку **ОК** в текущем диалоговом окне.

Также добавлять и удалять поддержку, конфигурировать FTP- и RDS-серверы можно непосредственно из ColdFusion Studio.



Интерфейс ColdFusion Studio

ColdFusion Studio предлагает удобный интерфейс. Здесь вы можете размещать и настраивать элементы рабочего пространства по своему усмотрению. Почти все элементы подвижны, так что перемещение различных панелей инструментов в удобное для вас местоположение не составляет труда. На рис. 3.1 показана область рабочего пространства ColdFusion Studio.

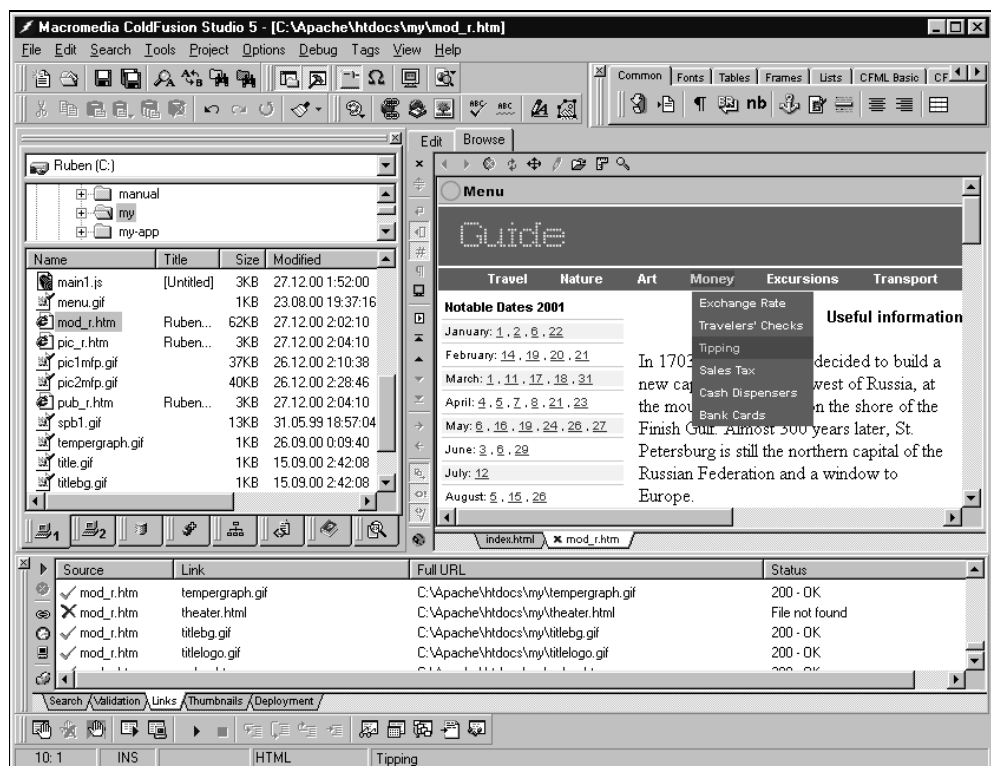


Рис. 3.1. Рабочее пространство ColdFusion Studio

Рабочее пространство ColdFusion Studio состоит из следующих элементов: главное меню, область для панелей инструментов, полоса быстрого доступа к набору инструментов, окно дополнительных возможностей, окно вывода результата, линейка инструментов режима редактирования, линейка инструментов режима просмотра и главное — это непосредственно область работы с документом.

Панели инструментов

В ColdFusion Studio панели инструментов могут размещаться в верхней, нижней, левой или правой частях рабочей области редактора, а также могут быть расположены как в отдельном окне, так и в специализированной полосе быстрого доступа к набору инструментов (**QuickBar**).

По умолчанию в верхней части рабочей области находятся четыре панели инструментов: стандартная панель, панель редактирования, панель режимов просмотра и панель вспомогательных средств. Давайте рассмотрим перечисленные панели более подробно.

Панель инструментов **Standard** (Стандартная) включает следующие инструменты (рис. 3.2):

- ☐ **New Document** (Создать) — создает новый документ;
- ☐ **Open** (Открыть) — открывает существующий документ;
- ☐ **Save** (Сохранить) — сохраняет изменения в текущем документе;
- ☐ **Save All** (Сохранить все) — сохраняет изменения во всех открытых документах;
- ☐ **Find** (Найти) — осуществляет поиск фрагмента текста в текущем документе;
- ☐ **Replace** (Заменить) — заменяет фрагмент текста в текущем документе;
- ☐ **Extended Find** (Расширенный поиск) — осуществляет расширенный поиск в тексте документа;
- ☐ **Extended Replace** (Расширенная замена) — выполняет расширенную замену текста в нескольких файлах или всех файлах каталога.



Рис. 3.2. Панель инструментов **Standard**

Панель инструментов **Edit** (Правка) содержит следующие инструменты (рис. 3.3):

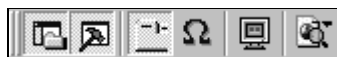
- ☐ **Cut** (Вырезать) — вырезает выделенный фрагмент в буфер обмена;
- ☐ **Copy** (Копировать) — копирует выделенный фрагмент в буфер обмена;

- ☐ **Paste** (Вставить) — вставляет фрагмент из буфера обмена;
- ☐ **Show clipboard** (Показать буфер обмена) — отображает содержимое буфера обмена для выборочной вставки;
- ☐ **Paste All** (Вставить все) — вставляет все фрагменты из буфера обмена;
- ☐ **Clear Clipboard** (Очистить) — освобождает буфер обмена;
- ☐ **Undo** (Отменить) — отменяет последнее действие;
- ☐ **Redo** (Повторить) — повторно выполняет команду;
- ☐ **Repeat Last Tag** (Повторить последний тег) — повторяет ввод последнего набранного тега;
- ☐ **CodeSweeper** (Автоформат) — выполняет функцию автоматизированного процесса форматирования кода.

Рис. 3.3. Панель инструментов **Edit**

Панель инструментов **View** (Просмотр) включает следующие инструменты (рис. 3.4):

- ☐ **Resource Tab** (Вкладки ресурсов) — показывает/скрывает панель ресурсов;
- ☐ **Results** (Результаты) — показывает/скрывает окно результатов;
- ☐ **QuickBar** (Быстрый доступ) — показывает/скрывает полосу быстрого доступа к набору инструментов;
- ☐ **Extended Characters** (Специальные символы) — показывает/скрывает набор специальных символов;
- ☐ **Full Screen** (Полный экран) — выводит документ на весь экран;
- ☐ **View External Browser List** (Список внешних браузеров) — показывает результат проектирования документа в одном из предустановленных браузеров.

Рис. 3.4. Панель инструментов **View**

Панель вспомогательных средств **Tools** (Сервис) включает следующие инструменты (рис. 3.5):

- ☐ **Palette** (Палитра) — вызывает визуальный редактор цветов;
- ☐ **Verify Links** (Проверить ссылки) — производит проверку ссылок;

- ☐ **Validate Current Document** (Допустимость текущего документа) — проверяет правильность написания текущего документа в зависимости от указанных значений в настройках среды выбором команды **Options | Settings | Validation** (Настройка | Параметры | Допустимость);
- ☐ **Thumbnails** (Миниатюры) — демонстрирует графические файлы из текущего каталога;
- ☐ **Spell Check** (Орфография) — запускает процесс проверки орфографии;
- ☐ **Mark Spelling Errors** (Проверять орфографию в режиме редактирования) — включает/выключает режим выделения ошибок;
- ☐ **Style Editor** (Редактор стилей) — запускает редактор каскадных таблиц стилей TopStyle;
- ☐ **New Image Map** (Новое изображение-карта) — создает графические ссылки с разбивкой на части.



Рис. 3.5. Панель инструментов **Tools**

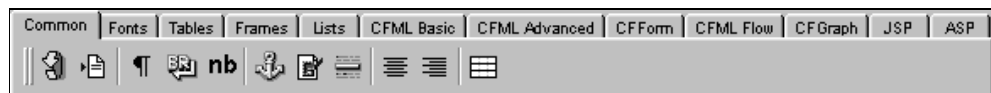
Полоса быстрого доступа к наборам инструментов

QuickBar (Быстрый доступ) — особая панель инструментов с вкладками, предназначенная для ускорения написания кода. Количество вкладок соответствует числу включенных в эту полосу панелей инструментов. По умолчанию полоса набора инструментов, о которой идет речь, включает панели, перечисленные ниже.

Общая панель **Common** (Общие) содержит следующие инструменты (рис. 3.6):

- ☐ **Quick start** (Быстрый запуск) — запускает мастер для быстрого создания документов;
- ☐ **Body** (<Body>) — вызывает диалоговое окно редактирования тега тела документа;
- ☐ **Paragraph** (Параграф) — вставляет парные теги начала и окончания абзаца;
- ☐ **Break** (Разрыв строки) — вставляет тег перевода строки;
- ☐ **Non-breaking Space** (Неразрывный пробел) — вставляет символ пробела в виде escape-последовательности;
- ☐ **Anchor** (Якорь) — открывает диалоговое окно редактирования тега гиперсвязи документа;

- ☐ **Image** (Изображение) — вызывает диалог редактирования тега графического изображения;
- ☐ **Horizontal Rule** (Горизонтальная линейка) — открывает диалоговое окно редактирования тега горизонтальной линии;
- ☐ **Align Center** (По центру) — вставляет логический блок с выравниванием по центру;
- ☐ **Align Right** (По правому краю) — выравнивает абзац по правому краю;
- ☐ **DIV** (<DIV>) — открывает диалог редактирования тега слоя.

Рис. 3.6. Панель **Common**

Панель шрифтов **Fonts** (Шрифты) включает следующие инструменты (рис. 3.7):

- ☐ **Font** (Шрифт) — открывает диалог редактирования тега формата шрифта;
- ☐ **Font Size +** (Размер +) — окружает выделенный блок тегами, позволяющими задать размер шрифта данного блока равным +1;
- ☐ **Font Size -** (Размер -) — окружает выделенный блок тегами, позволяющими определить размер шрифта данного блока равным -1;
- ☐ **Bold** (Полужирный) — окружает выделенный блок тегами, позволяющими определить характер шрифта данного блока, как полужирный;
- ☐ **Italic** (Курсив) — окружает выделенный блок тегами, позволяющими определить характер шрифта данного блока, как курсивный;
- ☐ **Strong** (Жирный) — окружает выделенный блок тегами, позволяющими задать характер шрифта блока, как жирный;
- ☐ **Emphasis** (Наклонный) — окружает выделенный блок тегами, позволяющими определить характер шрифта данного блока, как курсивный;
- ☐ **Preformatted Text** (Предварительно отформатированный) — открывает диалог редактирования тега, позволяющего определить характер текста, отображаемого на экране "как есть", с сохранением пользовательских пробелов, переносов строк и отступов табуляции;
- ☐ **Heading 1** (Заголовок 1) — окружает выделенный блок тегами, позволяющими определить характер данного блока, как заголовок первого уровня;
- ☐ **Heading 2** (Заголовок 2) — окружает выделенный блок тегами, позволяющими определить характер данного блока, как заголовок второго уровня;

- ☐ **Heading 3** (Заголовок 3) — окружает выделенный блок тегами, позволяющими определить характер данного блока, как заголовок третьего уровня;
- ☐ **Subscript** (Подстрочный) — окружает выделенный блок тегами, позволяющими определить характер данного блока, как подстрочный текст;
- ☐ **Superscript** (Надстрочный) — окружает выделенный блок тегами, позволяющими определить характер данного блока, как надстрочный текст.

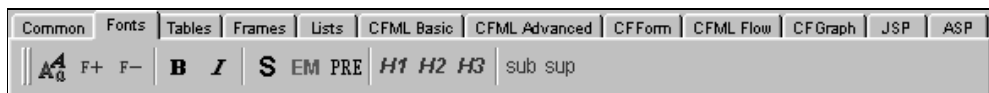
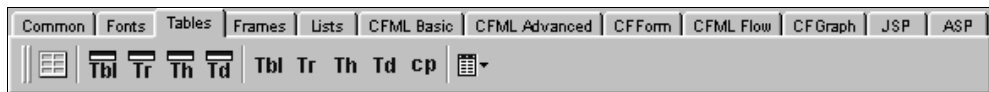


Рис. 3.7. Панель Fonts

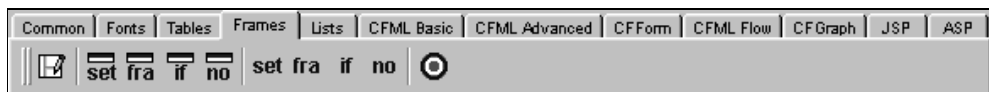
Панель таблиц **Tables** (Таблицы) предоставляет следующие инструменты (рис. 3.8):

- ☐ **Table Wizard** (Мастер таблиц) — запускает мастер для быстрого создания таблиц;
- ☐ **Table Dialog** (Изменить таблицу) — вызывает диалог редактирования тега таблицы;
- ☐ **Table Row Dialog** (Изменить строку) — открывает диалоговое окно редактирования тега строки таблицы;
- ☐ **Table Header Dialog** (Изменить шапку таблицы) — открывает диалоговое окно редактирования тега ячейки таблицы, играющей роль заголовка колонки;
- ☐ **Table Data Dialog** (Изменить данные) — вызывает диалог редактирования тега ячейки таблицы;
- ☐ **Table** (Таблица) — вставляет пару тегов начала и окончания таблицы;
- ☐ **Table Row** (Строка) — добавляет пару тегов начала и окончания строки таблицы;
- ☐ **Table Header** (Шапка таблицы) — вставляет пару тегов, начала и окончания ячейки, играющей роль заголовка колонки;
- ☐ **Table Data** (Данные) — размещает пару тегов начала и окончания ячейки таблицы;
- ☐ **Table Caption** (Заголовок таблицы) — вставляет пару тегов, определяющих заголовок таблицы;
- ☐ **Table Sizer — Quick Table** (Размер таблицы) — вставляет блок тегов таблицы с использованием окна, позволяющего задать количество строк и колонок создаваемой таблицы.

Рис. 3.8. Панель **Tables**

Панель фреймов **Frames** (Фреймы) включает следующие инструменты (рис. 3.9):

- ☐ **Frame wizard** (Мастер фреймов) — запускает мастер для быстрого создания фреймов;
- ☐ **Frameset Dialog** (Изменить набор фреймов) — открывает диалог редактирования тега контейнера фреймов;
- ☐ **Frame Dialog** (Изменить фрейм) — отображает диалоговое окно редактирования тега фрейма;
- ☐ **Floating Frame Dialog** (Изменить плавающий фрейм) — вызывает диалоговое окно редактирования тега плавающего фрейма;
- ☐ **Noframes Dialog** (Изменить настройку "Без фреймов") — открывает диалог редактирования тега отсутствия поддержки фреймов;
- ☐ **Frameset** (Набор фреймов) — вставляет пару тегов начала и окончания набора фреймов;
- ☐ **Frames** (Фрейм) — добавляет тег начала фрейма;
- ☐ **Floating Frame** (Плавающий фрейм) — размещает пару тегов начала и окончания плавающего фрейма;
- ☐ **No Frames** (Без фреймов) — вставляет пару тегов для определения блока, предназначенного для браузеров, не поддерживающих фреймы;
- ☐ **Base Target** (Базовая цель) — открывает диалоговое окно редактирования тега, позволяющего жестко устанавливать полный URL-адрес сайта в качестве базового.

Рис. 3.9. Панель **Frames**

Панель списков **Lists** (Списки) включает следующие инструменты (рис. 3.10):

- ☐ **Quick List** (Список) — открывает диалог формирования списка;
- ☐ **Unordered List** (Маркированный список) — отображает диалог редактирования тега неупорядоченного маркированного списка;
- ☐ **Ordered List** (Нумерованный список) — вызывает диалог редактирования тега упорядоченного списка;

- ❑ **List Item** (Элемент списка) — открывает диалог редактирования тега, определяющего пункт списка;
- ❑ **Definition List** (Список определений) — вставляет теги для списка определений;
- ❑ **Definition Term** (Термин списка определений) — вызывает диалоговое окно редактирования тега термина списка определений;
- ❑ **Definition** (Определение) — отображает диалоговое окно редактирования тега вхождения списка определений;
- ❑ **Menu** (Меню) — вставляет пару тегов начала и окончания списка меню.

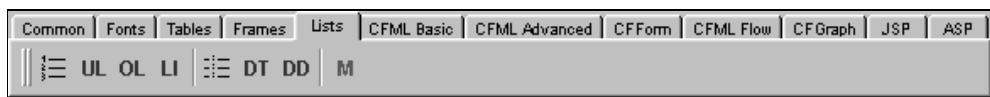
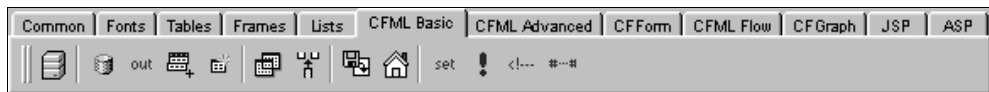


Рис. 3.10. Панель **Lists**

Панель **CFML Basic** (Основные теги CFML) содержит следующие инструменты (рис. 3.11):

- ❑ **ColdFusion Server Variables** (Переменные ColdFusion Server) — открывает диалог построителя выражений ColdFusion;
- ❑ **CFQUERY** — вставляет тег `<CFQUERY>` для формирования запроса;
- ❑ **CFOUTPUT** — вставляет тег `<CFOUTPUT>`, позволяющий осуществлять вывод на экран;
- ❑ **CFINSERT** — размещает тег `<CFINSERT>` добавления записи через источник данных;
- ❑ **CFUPDATE** — вставляет тег `<CFUPDATE>` для обновления записи через источник данных;
- ❑ **CFTABLE** — размещает тег `<CFTABLE>` создания таблиц;
- ❑ **CFCOL** — вставляет тег `<CFCOL>`, определяющий параметры колонок таблиц;
- ❑ **CFINCLUDE** — добавляет тег `<CFINCLUDE>` для включения содержимого файла;
- ❑ **CFLOCATION** — вставляет тег `<CFLOCATION>` открытия ресурса;
- ❑ **CFSET** — вставляет тег `<CFSET>` определения переменной;
- ❑ **CFPARAM** — помещает тег `<CFPARAM>` для определения переменной с исходным значением;
- ❑ **ColdFusion Comment** (Комментарий ColdFusion) — добавляет пару тегов для определения комментария;
- ❑ **Surround with #** (Определение переменных) — вставляет пару тегов для определения переменных.

Рис. 3.11. Панель **CFML Basic**

Панель **CFML Advanced** (Дополнительные теги CFML) включает следующие инструменты (рис. 3.12):

- ☐ **CFCOOKIE** — добавляет тег `<CFCOOKIE>` для определения cookies;
- ☐ **CFCONTENT** — вставляет тег `<CFCONTENT>`, позволяющий включать содержимое файлов в текущую страницу;
- ☐ **CFHEADER** — добавляет тег `<CFHEADER>` для генерация HTTP-заголовков;
- ☐ **CFAPPLICATION** — помещает тег `<CFAPPLICATION>` для определения механизма хранения переменных приложения;
- ☐ **CFERROR** — добавляет тег `<CFERROR>` определения демонстрации пользовательской страницы при возникновении ошибок;
- ☐ **CFDIRECTORY** — вставляет тег `<CFDIRECTORY>`, обеспечивающий работу с каталогами;
- ☐ **CFFILE** — добавляет тег `<CFFILE>`, обеспечивающий работу с файлами;
- ☐ **CFMAIL** — помещает тег `<CFMAIL>` для отправки писем по электронной почте;
- ☐ **CFPOP** — добавляет тег `<CFPOP>` для получения электронной почты;
- ☐ **CFHTTP** — добавляет тег `<CFHTTP>`, обеспечивающий работу с Web-узлами;
- ☐ **CFHTTPPARAM** — вставляет тег `<CFHTTPPARAM>` с целью определения параметров для операций метода `<CFHTTP POST>`;
- ☐ **CFLDAP** — добавляет тег `<CFLDAP>`, обеспечивающий доступ к LDAP-серверу;
- ☐ **CFFTP** — добавляет тег `<CFFTP>`, обеспечивающий доступ к FTP-серверу;
- ☐ **CFSEARCH** — вставляет тег `<CFSEARCH>` для выполнения поиска индексированных данных;
- ☐ **CFINDEX** — размещает тег `<CFINDEX>`, определяющий коллекцию для индексации данных;

Рис. 3.12. Панель **CFML Advanced**

❑ **CFMODULE** — добавляет тег `<CFMODULE>`, активизирующий пользовательский тег;

❑ **CFOBJECT** — вставляет тег `<CFOBJECT>` для создания объекта.

Панель **CF Form** (CF-теги для построения форм) включает следующие инструменты (рис. 3.13):

❑ **CFFORM** — вставляет тег `<CFFORM>` для построения формы ввода на стороне клиента;

❑ **CFAPPLET** — добавляет тег `<CFAPPLET>`, запускающий Java-апплет;

❑ **CFINPUT Text** — вставляет тег `<CFINPUT Text>` создания элемента формы "поле ввода";

❑ **CFINPUT Radio** — помещает тег `<CFINPUT Radio>` для создания элемента формы "переключатель";

❑ **CFINPUT Checkbox** — вставляет тег `<CFINPUT Checkbox>`, создающий элемент формы "флажок";

❑ **CFSELECT** — добавляет тег `<CFSELECT>` для создания элемента формы "список";

❑ **CFTREE** — вставляет тег `<CFTREE>` для создания управляющего элемента "дерево" (Tree);

❑ **CFTREEITEM** — помещает тег `<CFTREEITEM>`, определяющий пункт управляющего элемента "дерево";

❑ **CFGRID** — вставляет тег `<CFGRID>`, создающий управляющий элемент "сетка" (Grid);

❑ **CFGRIDCOLUMN** — добавляет тег `<CFGRIDCOLUMN>`, определяющий столбцы управляющего элемента "сетка";

❑ **CFGRIDROW** — размещает тег `<CFGRIDROW>`, определяющий строки управляющего элемента "сетка";

❑ **CFGRIDUPDATE** — вставляет тег `<CFGRIDUPDATE>` для обновления отредактированных данных с помощью "сетки";

❑ **CFSLIDER** — добавляет тег `<CFSLIDER>`, определяющий элемент "ползунок" (Slider).

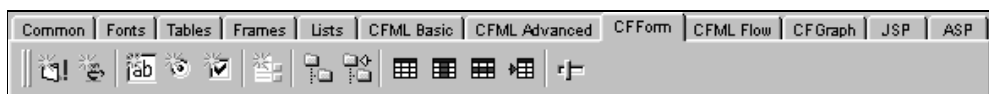


Рис. 3.13. Панель **CF Form**

Панель **CFML Flow** (Операторы CFML) включает инструменты (рис. 3.14):

❑ **CFTRY** — вставляет тег `<CFTRY>` для обработки исключительных ситуаций; оператор `try`;

- ❑ **CFCATCH** — добавляет тег <CFCATCH> для обработки исключительных ситуаций; оператор `catch`;
- ❑ **CFTHROW** — помещает тег <CFTHROW> для обработки исключительных ситуаций; оператор `throw`;
- ❑ **CFLOCK** — вставляет тег <CFLOCK> для обеспечения поддержки целостности данных и синхронизации CFML-кода;
- ❑ **CFSWITCH** — помещает тег <CFSWITCH>; условный оператор `switch`;
- ❑ **CFCASE** — вставляет тег <CFCASE>; условный оператор `case`;
- ❑ **CFDEFAULTCASE** — добавляет тег <CFDEFAULTCASE>; условный оператор `defaultcase`;
- ❑ **CFSCRIPT** — вставляет тег <CFSCRIPT>, определяющий блок кода CFScript;
- ❑ **CFIF** — помещает тег <CFIF>; условный оператор `if`;
- ❑ **CFELSE** — добавляет тег <CFELSE>; условный оператор `else`;
- ❑ **CFELSEIF** — вставляет тег <CFELSEIF>; условный оператор `elseif`;
- ❑ **CFBREAK** — помещает тег <CFBREAK>, осуществляющий переход из повторяющегося набора инструкций, игнорируя условия;
- ❑ **CFABORT** — вставляет тег <CFABORT>, приостанавливающий процесс;
- ❑ **CFEXIT** — добавляет тег <CFEXIT>, осуществляющий обработку аварийного прекращения выполнения CFML-тегов;
- ❑ **CFLOOP** — помещает тег <CFLOOP>, обозначающий повторение набора инструкций, основанных на условиях.

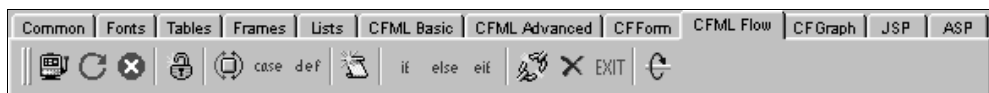


Рис. 3.14. Панель CFML Flow

Панель **CF Graph** (CF-теги для построения диаграмм) включает инструменты (рис. 3.15):

- ❑ **CFGRAPH Bar** — помещает тег <CFGRAPH> для создания диаграммы в виде столбцов (гистограмма);
- ❑ **CFGRAPH HorizontalBar** — помещает тег <CFGRAPH> для создания диаграммы в виде горизонтальных столбцов (линейчатая диаграмма);
- ❑ **CFGRAPH Line** — помещает тег <CFGRAPH> для создания диаграммы в виде линий (график);

- ☐ **CFGRAPH Pie** — помещает тег <CFGRAPH> для создания круговой диаграммы.

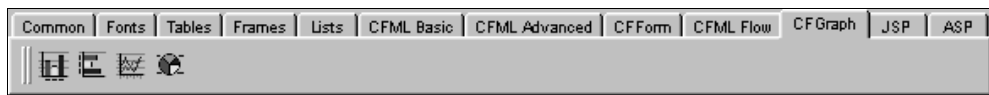


Рис. 3.15. Панель **CF Graph**

Панель **JSP** (Java Server Pages) включает инструменты (рис. 3.16):

- ☐ **Page** — позволяет включать директиву `page` для движка сервлета (Servlet) с информацией об основных настройках JSP;
- ☐ **Include** — добавляет строку кода для определения файла в качестве включаемого;
- ☐ **TagLib** — определяет пользовательскую библиотеку тегов для текущей JSP-страницы;
- ☐ **Declaration** — добавляет пару тегов `<% %>` для объявления переменных JSP;
- ☐ **Scriptlet** — добавляет пару тегов `<% %>` для использования внутри этих тегов JSP-скрипта;
- ☐ **Expression** — добавляет пару тегов `<%= %>` для обработки JSP-выражения и выводит полученный результат на экран;
- ☐ **jsp:useBean** — определяет экземпляр Java Bean;
- ☐ **jsp:setProperty** — присваивает значения свойствам ранее описанных beans;
- ☐ **jsp:getProperty** — определяет значение свойства bean-компонента, конвертирует его в строку и направляет в поток вывода;
- ☐ **jsp:include** — вызывает одну JSP-страницу из другой, после завершения выполнения вызванной страницы контроль управления передается вызывающей JSP-странице;
- ☐ **jsp:forward** — вызывает одну JSP-страницу из другой, после завершения выполнения вызванной страницы контроль управления не передается вызывающей JSP-странице;
- ☐ **jsp:params** — добавляет пару тегов `<jsp:params></jsp:params>` для определения параметров;
- ☐ **jsp:param** — предназначен для определения параметра при использовании JSP-тега;
- ☐ **jsp:plugin** — вставляет элемент `OBJECT` или `EMBED` (в зависимости от типа браузера), необходимый для запуска апплетов, использующих plug-ins Java;
- ☐ **jsp Comment** — добавляет пару тегов `<%-- --%>`, используемых в качестве комментария в JSP.



Рис. 3.16. Панель JSP

Панель **ASP** (Active Server Pages) включает инструменты (рис. 3.17):

- ☐ **ASP Server Variables** (Переменные ASP Server) — позволяет включать серверные переменные ASP;
- ☐ **ASP Include** — добавляет строку кода для определения файла в качестве включаемого;
- ☐ **ASP Tag** — добавляет пару тегов `<% %>` для использования внутри них ASP-кода;
- ☐ **ASP Output** — добавляет пару тегов `<%= %>` для обработки ASP-выражения и выводит полученный результат на экран;
- ☐ **ASP If** — добавляет строку кода для реализации условного оператора `if`;
- ☐ **ASP Else** — добавляет строку кода для реализации условного оператора `else`;
- ☐ **ASP End** — добавляет строку кода для реализации условного оператора `end`.

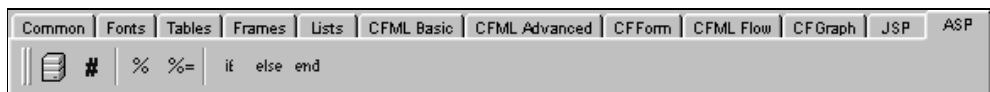


Рис. 3.17. Панель ASP

Мы не задаемся целью подробно рассмотреть каждый из перечисленных инструментов. Отметим лишь, что предназначение инструментов, расположенных на таких панелях, как: **CFML Basic** (Основные теги CFML), **CFML Advanced** (Дополнительные теги CFML), **CF Form** (CF-теги для построения форм), **CFML Flow** (Операторы CFML) и **CF Graph** (CF-теги для построения диаграмм) сводится к одному — вставке того или иного тега. Большинство этих инструментов предполагает редактирование атрибутов выбранного тега в режиме диалога. В качестве примера разберем инструмент **CFMAIL**, расположенный на панели инструментов **CFML Advanced** (Дополнительные теги CFML) и предназначенный для вставки тега `<CFMAIL>`, способствующего отправке писем по электронной почте. На рис. 3.18 изображено диалоговое окно редактирования атрибутов указанного тега.

Обратите внимание на флажок **Output attributes on single line** (Выводить атрибуты в одну строку), который позволяет указывать правило размещения кода: по умолчанию код располагается в одну строку. Автор предпочитает снимать флажок для придания коду более читабельного вида, т. к. в этом

случае каждый атрибут тега размещается с новой строки с соблюдением определенных отступов. На рис. 3.19 изображено диалоговое окно редактирования сообщения тега <CFMAIL>.

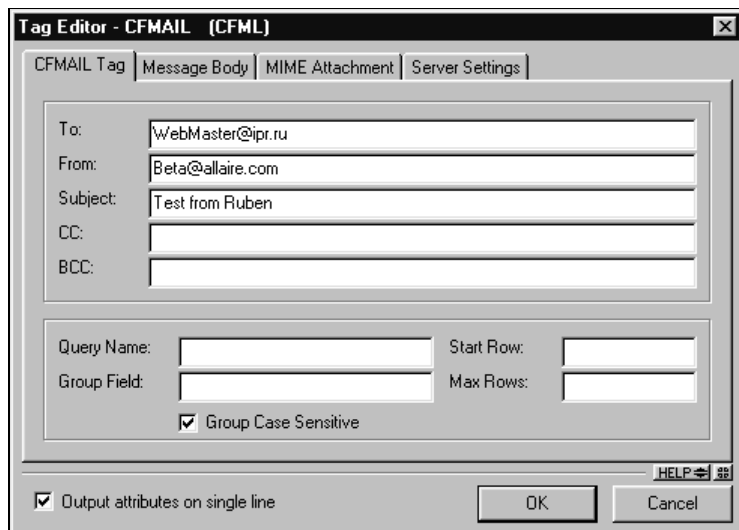


Рис. 3.18. Диалоговое окно **Tag Editor - CFMAIL (CFML)** для редактирования атрибутов тега <CFMAIL>

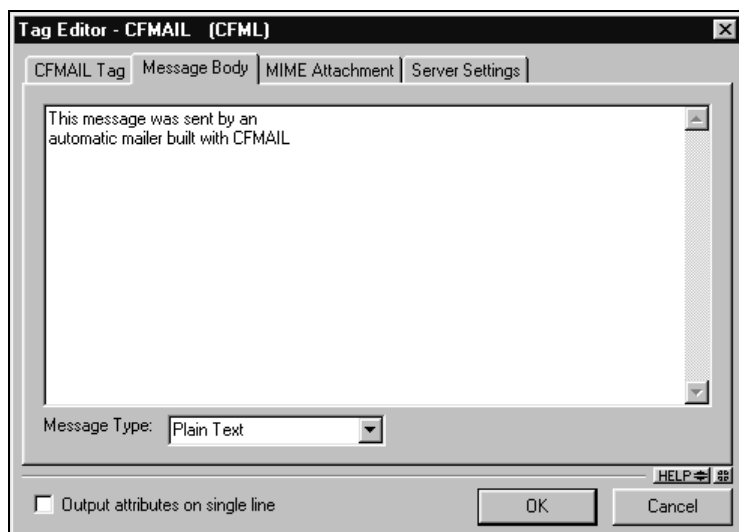


Рис. 3.19. Диалоговое окно **Tag Editor - CFMAIL (CFML)**, вкладка **Message Body**

Здесь помимо текста сообщения можно указать его тип (в раскрывающемся списке **Message Type** (Тип сообщения)), который принимает значения явного текста или HTML-кода.

При необходимости прикрепления файлов к отправляемому письму воспользуйтесь вкладкой **MIME Attachment** (Присоединение MIME) диалогового окна **Tag Editor - CFMAIL (CFML)** (Редактор тегов — CFMAIL (CFML)). На вкладке **Server Settings** (Параметры сервера) не забудьте указать ваш почтовый сервер. На рис. 3.20 показан код, полученный в результате изложенных выше действий.

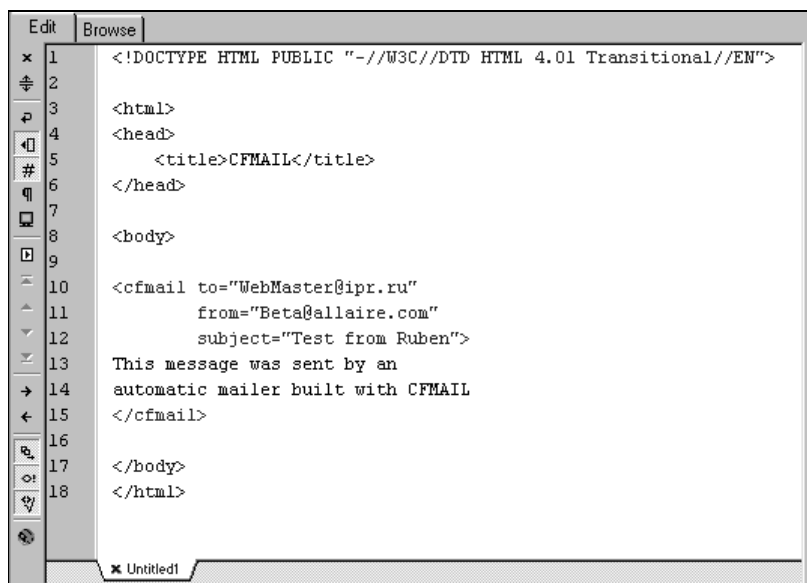


Рис. 3.20. Результат работы инструмента **CFMAIL**

В подобных случаях ColdFusion Studio — достаточно удобное средство, если вы забыли синтаксис какого-либо тега, выбрали инструмент вставки тега, ввели значения необходимых атрибутов, подтвердили свои действия нажатием кнопки **ОК**. А далее редактируйте полученный код по мере необходимости. Что еще надо? Тем более, все теги ColdFusion¹ поддерживаются средством автоматической простановки тегов, так называемыми *подсказками*, о чем шла речь в главе 2.

Но и это еще не все, каждое диалоговое окно редактирования включает кнопки помощи (табл. 3.1).

¹ Вместо ColdFusion может встречаться сокращение CF. — *Ред.*

Таблица 3.1. Кнопки помощи в диалоговых окнах редактирования

Кнопка	Описание
	Кнопка с надписью Help (Справка) позволяет в нижней части диалогового окна отобразить вспомогательный текст
	Кнопка, напоминающая перекрестие, позволяет открыть в новом окне текст помощи со сведениями о текущем инструменте

Режимы работы с документами

ColdFusion Studio предполагает три режима работы с документами:

- ☐ **Edit** (Редактор) — рабочая область для редактирования документа;
- ☐ **Browse** (Просмотр) — просмотр документов в браузере, который установлен по умолчанию;
- ☐ **Help** (Справка) — режим просмотра вспомогательной информации (для пятой версии), или **Design** (Проектирование) — режим визуального редактирования документов (заменен в Macromedia ColdFusion Studio 5 Release Candidate 1 на режим **Help**).

На рис. 3.21 изображен набор вкладок для перехода между режимами работы **Edit** (Редактор), **Browse** (Просмотр) и **Help** (Справка) (опять же, это применительно к версии 5).

**Рис. 3.21.** Набор вкладок для перехода по режимам работы

Режим помощи появляется только при работе с соответствующими документами. Переход между двумя режимами **Edit** (Редактор) и **Browse** (Просмотр) можно выполнять посредством функциональной клавиши <F12>, переход между режимами **Edit** (Редактор) и **Help** (Справка) — осуществляется с использованием комбинации клавиш <Shift>+<F12>. Для более ранних версий эта комбинация применялась для перехода между режимами **Edit** (Редактор) и **Design** (Проектирование).

Как для режима **Edit** (Редактор), так и для режима **Browse** (Просмотр) существует собственная панель инструментов, которая способствует облегчению работы с документами. При этом в режиме **Help** (Справка) дублируется часть инструментов из режима **Browse** (Просмотр).

Панель инструментов режима редактирования документов *Edit*

Панель инструментов режима редактирования документов (**Editor Toolbar**) может быть скрыта или отображена командой главного меню **View | Editor Toolbar** (Вид | Панель инструментов режима редактирования) в зависимости от текущего состояния индикатора. Панель инструментов располагается вертикально, "аккуратно" прилегая слева к рабочей области редактора. В табл. 3.2 перечислены все инструменты полезной панели.

Таблица 3.2. Инструменты для режима редактирования документов *Edit*

Изображение	Название	Описание
	Close Active Document (Заккрыть активный документ)	Закрывает текущий документ
	Split Current Document (Разделить текущий документ)	Разделяет окно редактора на две части, что достаточно удобно для одновременного наблюдения за разными частями документа, находящимися друг от друга на значительном расстоянии
	Word Wrap (Перенос слов)	Включает/выключает режим контроля длины строк текста в окне редактора. В случае включения данного режима невидимый фрагмент правой части документа автоматически переносится на новую строку. Таким образом, текст выравнивается по ширине окна редактора, что упрощает работу с документами "невероятной" ширины. Отметим, что при этом сам документ не форматируется, стоит только отключить данный режим, и все вернется на свое место
	Show Gutter (Показать вспомогательную область)	Отображает/скрывает область с левой стороны документа, предназначенную для вывода нумерации строк, переносов, закладок и отслеживания точек прерывания
	Show Line Numbers in Gutter (Показать нумерацию строк)	Включает/выключает показ нумерации строк документа
	Show Hidden Characters (Показать скрытые символы)	Включает/выключает показ невидимых символов
	Full Screen (Полный экран)	Разворачивает документ на весь экран

Таблица 3.2 (окончание)

Изображение	Название	Описание
	Show Open Document List (Показать список открытых документов)	Отображает список открытых документов, демонстрируя полный путь размещения каждого с возможностью активизировать необходимый документ для дальнейшей работы с ним (рис. 3.22)
	First Document (Первый документ)	Осуществляет переход на первый документ
	Previous Document (Предыдущий документ)	Выполняет переход на предыдущий документ
	Next Document (Следующий документ)	Производит переход на следующий документ
	Last Document (Последний документ)	Осуществляет переход на последний документ
	Indent (Отступ)	Создает отступы внутри текущего документа
	Unindent (Выступ)	Совершает обратное действие предыдущему инструменту — убирает отступы
	Tag Insight (Список тегов)	Включает/выключает режим автоматической вставки тегов, так называемых <i>всплывающих подсказок</i> , при вводе зарегистрированных тегов
	Tag Completion (Завершение тега)	Включает/выключает режим автоматического завершения тегов. То есть стоит ввести первую часть тега, например <code><p></code> , как тут же появится вторая часть — <code></p></code> , а текстовый курсор будет помещен "внутри" приведенного тега. Таким образом, останется только ввести содержание, в данном случае, абзаца
	Tag Validation (Допустимость тега)	Включает/выключает режим проверки правильности ввода тегов. Суть этого режима достаточно проста: стоит ввести несуществующий тег, и вы тут же получите сообщение об ошибке. Что приятно, данные сообщения отражаются в строке состояния Status Bar окна ColdFusion Studio, а не в отдельном окне с "назойливой" кнопкой подтверждения, как это бывает у некоторых программных продуктов
	Macromedia Dreamweaver	Открывает текущий документ в редакторе Dreamweaver, предварительно предлагая сохранить все изменения, произведенные по отношению к этому документу

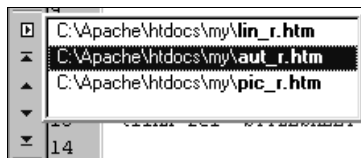


Рис. 3.22. Список открытых документов

Панель инструментов режима просмотра документов *Browse*

Панель инструментов **Browser Toolbar** режима просмотра документов в отличие от панели инструментов редактирования, располагается горизонтально над рабочей областью редактора (рис. 3.23).



Рис. 3.23. Панель инструментов режима просмотра документов

Перечислим все инструменты этой панели:

- ☐ **Previous** (Назад) — осуществляет переход на предыдущую страницу. Здесь подразумевается "поход" по посещенным страницам в режиме просмотра, аналогично браузеру;
- ☐ **Next** (Вперед) — выполняет переход на следующую страницу;
- ☐ **Stop** (Остановить) — позволяет остановить загружаемую страницу;
- ☐ **Refresh** (Обновить) — обновляет текущую страницу;
- ☐ **Browser Size** (Размер браузера) — изменяет размер браузера. Подразумевается три режима: 640×480, 800×600 и в размер окна;
- ☐ **Edit** (Правка) — позволяет редактировать документ;
- ☐ **Open URL** (Открыть URL) — открывает URL для редактирования;
- ☐ **Toggle Rulers** (Линейки размерности) — отображает линейку, позволяющую сопоставить размер элементов текущей страницы;
- ☐ **Open in External Browser** (Открыть во внешнем браузере) — открывает текущий документ во внешнем браузере, установленном по умолчанию.

Режим проектирования *Design*

ColdFusion Studio более ранних версий предоставлял дополнительную опцию для визуального редактирования кода — **Design** (Проектирование). Данная опция работает по принципу WYSIWYG ("What You See Is What You Get" —

"Что вы видите, то вы получаете"). Страница создается так, как она будет выглядеть в окне браузера, а программа записывает ее в виде HTML-кода.

Режим **Design** (Проектирование) может быть полезен при размещении изображений на странице и для создания комплексных таблиц и форм. Код для этих элементов генерируется автоматически и обновляется по мере появления каких-либо изменений.

Данный режим предполагает наличие браузера Microsoft Internet Explorer версии 4.01 или более поздней. Мы не рассматриваем панели инструментов для режима работы **Design** (Проектирование) в связи с отсутствием данного режима работы в новой версии — CF Studio 5.

В завершение рассказа о режиме проектирования отметим, что существует возможность вставки листов, таблиц и ячеек путем копирования из офисных пакетов MS Word и MS Excel. И при переходе в другой режим работы с документом программа преобразовывает содержимое рабочей страницы в эквивалентный формат HTML. Результат получается достаточно убедительный, и что немаловажно, код в этом случае хорошо читается.

Навигация по открытым документам

Мы уже познакомились с панелью инструментов режима редактирования документов, где существует, как минимум, два способа перехода по открытым документам, однако этого мало.

В каком бы режиме вы не находились — редактирования, просмотра или проектирования — в нижней части рабочей области чуть выше панели отладки документов (**Debugger Toolbar**) существует горизонтальная полоса с вкладками (ярлыками), отображающими имена открытых документов, очень похожая на ярлыки листов из MS Excel. На рис. 3.24 приведен пример данной полосы вкладок открытых документов.



Рис. 3.24. Горизонтальная полоса вкладок открытых документов

Наименования всех документов, подвергшихся каким-либо изменениям, выделяются синим цветом. После сохранения изменений, наименование документа вновь приобретает привычный черный цвет. Имя файла текущего документа выделено белым фоном. Заметьте, если не хватает пространства для перечисления документов, на данной полосе справа появляются две кнопки-стрелки, позволяющие перемещаться по списку документов. Если же вы открываете документ, используя удаленное соединение, то это также отражается на ярлыке данного документа, пометкой его зеленым индикатором.

Панель ресурсов

Панель ресурсов (**Resource Tab**) представляет собой вспомогательное средство по работе с файловой системой (**Files1**) и (**Files2**), базами данных (**Database**), проектами (**Projects**), обозревателем сайта (**Site View**), фрагментами кодов (**Snippets**), системой помощи (**Help**) и инспектором тегов (**Tag Inspector**). Так называемая панель ресурсов, которую можно было бы назвать и окном ресурсов, разбита на тематические разделы, доступ к которым можно получить через соответствующие вкладки (рис. 3.25).

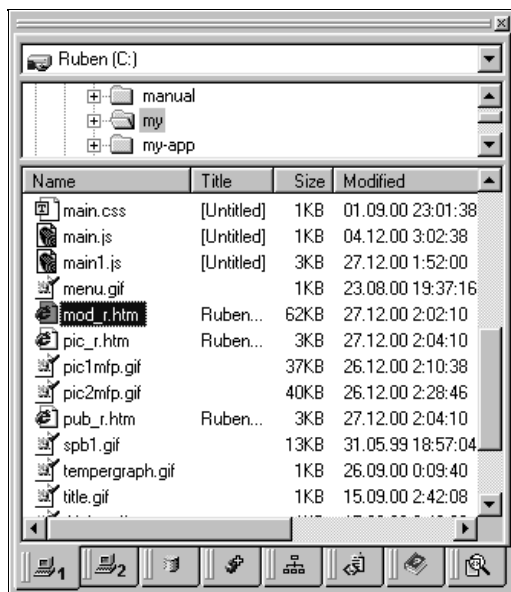


Рис. 3.25. Панель ресурсов **Resource Tab**

Панель ресурсов может быть скрыта с использованием команды главного меню **View | Resource Tab** (Вид | Панель ресурсов) или клавиши <F9>, с помощью этой же команды панель ресурсов раскрывается. Для изменения порядка отображения вкладок достаточно щелкнуть правой кнопкой мыши по необходимой вкладке и протящить ее вправо или влево в зависимости от ситуации. Более того, содержимое каждого тематического раздела можно разместить в отдельном окне. Для этого достаточно щелкнуть левой кнопкой мыши по вкладке и перетащить ее в сторону.

Рассмотрим по порядку размещения все вкладки.

- ❑ **Files1** (Файлы1) и **File2** (Файлы2) — система управления файлами и каталогами, обеспечивающая доступ ко всем файлам, включая сетевые ресурсы и удаленные соединения. Здесь вы можете добавлять, изменять и уда-

лять каталоги и файлы, отображать список файлов с сортировкой, предварительно просматривать графические файлы, создавать список избранных каталогов для быстрого доступа. Достаточно часто приходится работать с проектами из разных источников, для облегчения перехода от проекта к проекту разработчики включили в пятую версию две вкладки по работе с файлами, что действительно позволяет легко ориентироваться при работе над очередным "шедевром".

- ❑ **Database** (Базы данных) — обеспечивает доступ к базам данных через поименованные источники данных. Здесь вы можете просматривать данные, формировать запросы, сохранять запросы в качестве заготовок для повторного использования.
- ❑ **Projects** (Проекты) — предоставляет возможность размещения всех необходимых файлов для создания Web-сайта в легко доступном месте в рамках проекта.
- ❑ **Site View** (Обозреватель сайта) — осуществляет наглядное отображение всех связей текущего документа с другими файлами.
- ❑ **Snippets** (Фрагменты кодов) — обеспечивает работу с пользовательскими фрагментами кодов для многочисленного использования.
- ❑ **Tag Inspector** (Инспектор тегов) — поддерживает формирование дерева тегов без текста документа, что упрощает создание и редактирование тегов.
- ❑ **Help** (Справка) — система помощи, обеспечивающая доступ к справочнику по ColdFusion.

Описание работы с данными ресурсами читайте в *главе 4*.

Окно вывода результатов

Данное окно предназначено для инициализации просмотра результатов проверки и открывается каждый раз, когда выполняется команда тестирования текущего документа. В данном контексте подразумеваются такие команды, как **Tools | Validate Document** (Сервис | Допустимость документа), **Tools | Verify Links** (Сервис | Проверка ссылок).

Окно вывода результатов снабжено следующими вкладками (ярлыками):

- ❑ **Search** (Поиск) — отражает результат расширенного поиска (**Extended Find**) и расширенной замены (**Extended Replace**);
- ❑ **Validation** (Проверка) — демонстрирует список ошибок и предупреждений с комментариями, в зависимости от настроек спецификаций, указанных с помощью команды **Options | Settings | Validation** (Настройка | Параметры | Допустимость);
- ❑ **Links** (Ссылки) — отражает список связей текущего документа, предоставляя возможность документировать результат тестирования;

- ❑ **Thumbnails** (Миниатюры) — показывает все графические файлы из текущего каталога, предоставляя возможность с легкостью помещать графические изображения в разрабатываемые страницы;
- ❑ **Deployment** (Развертывание) — представляет информацию относительно проектируемого файла.

На рис. 3.26 изображено окно вывода результатов тестирования, открытое на вкладке **Links** (Ссылки).

Source	Link	Full URL	Status
✓ mod_r.htm	tempergraph.gif	C:\Apache\htdocs\my\tempergraph.gif	200 - OK
✗ mod_r.htm	theater.html	C:\Apache\htdocs\my\theater.html	File not found
✓ mod_r.htm	titlebg.gif	C:\Apache\htdocs\my\titlebg.gif	200 - OK
✓ mod_r.htm	titlelogo.gif	C:\Apache\htdocs\my\titlelogo.gif	200 - OK
✓ mod_r.htm	zak_r.htm	C:\Apache\htdocs\my\zak_r.htm	200 - OK
✗ mod_r.htm	zoo.html	C:\Apache\htdocs\my\zoo.html	File not found

Рис. 3.26. Окно вывода результатов **Results Window**

Трудно что-либо добавить по рассмотренному окну, заметим еще, что только со временем понимаешь, насколько удобно данное средство.

Панель отладки документов

В процессе разработки CFML-страниц мы сталкиваемся с ситуацией, когда программа работает не так, как задумывалось, или не работает вовсе. Даже опытному специалисту не всегда удастся избежать ошибок. На помощь может прийти средство отладки — панель **Debugger Toolbar** (Панель отладки) (рис. 3.27).

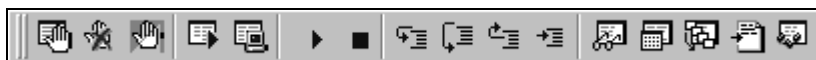


Рис. 3.27. Панель отладки **Debugger Toolbar**

Итак, по порядку размещения инструментов:

- ❑ **Breakpoints** (Точки прерывания) — отображает список точек прерывания;
- ❑ **Clear All Breakpoints** (Удалить все точки прерывания) — удаляет все точки прерывания;
- ❑ **Toggle Breakpoint** (Переключить точку прерывания) — переключает состояния точек прерывания;
- ❑ **Debug Start Settings** (Параметры запуска отладки) — осуществляет настройку параметров отладки документа;

- ❑ **Development Mappings** (Разработка карт) — выполняет настройку логических псевдонимов путей к файлам;
- ❑ **Start/Continue** (Начать/Продолжить) — начинает/продолжает отладку документа;
- ❑ **End Debug Session** (Завершить отладку) — прекращает отладку документа;
- ❑ **Step Into** (Шаг внутри) — выполняет следующую строку кода;
- ❑ **Step Over** (Шаг выше) — выполняет следующую строку кода, и если эта строка вызывает включенный файл (используя теги `<CFINCLUDE>`, `<CFMODULE>` или `<CFLOCATION>`), то вызываемый модуль выполняется в фоновом режиме;
- ❑ **Step Out** (Шаг наружу) — продолжает выполнение кода в блоке без остановки на каждой строке до тех пор, пока не будет достигнута строка, следующая за той, которая вызвала выполнение блока;
- ❑ **Run to Cursor** (Запустить до курсора) — выполняет код до строки, на которой находится курсор или точка прерывания. При этом положение курсора воспринимается как точка прерывания;
- ❑ **Watch** (Просмотр) — открывает окно отладки **Debug** (Отладка) на вкладке просмотра значений выражений, которые они принимают во время работы программы;
- ❑ **Recordsets** (Набор записей) — открывает окно отладки **Debug** (Отладка) на вкладке просмотра списка набора данных;
- ❑ **Stack** (Стек) — открывает окно отладки **Debug** (Отладка) на вкладке просмотра выполняемого тега;
- ❑ **Output** (Вывод) — открывает окно отладки **Debug** (Отладка) на вкладке просмотра результата выполнения активной страницы;
- ❑ **Variables** (Переменные) — открывает окно отладки **Debug** (Отладка) на вкладке просмотра списка локальных переменных.

Пример использования панели отладки см. в разд. "Отладка Web-страниц" гл. 4.

Клавиатурные комбинации для быстрого доступа

Как и в любом программном продукте, в ColdFusion Studio существует набор клавиатурных комбинаций для быстрого доступа к командам. В ColdFusion Studio есть возможность редактировать, а также добавлять собственные комбинации. В табл. 3.3 приведены клавиатурные комбинации при настройках по умолчанию.

Таблица 3.3. Клавиатурные комбинации при настройках по умолчанию

Клавиша или комбинация клавиш	Описание
<F1>	Справка
<F2>	Показать синтаксис текущего тега
<F3>	Найти
<F4>	Активизировать инспектор тегов (Tag Inspector)
<F6>	Проверить выделенный тег
<F7>	Проверить документ на орфографические ошибки
<F8>	Вызвать диалоговое окно настройки параметров
<F9>	Включить/выключить панель ресурсов (Resource Tab)
<F10>	Выйти на главное меню
<F11>	Просмотреть страницу в браузере, установленном по умолчанию
<F12>	Осуществить переход между двумя режимами: редактирование и просмотр
<ALT>+<F5>	Закончить отладку документа
<ALT>+<F6>	Удалить точки прерывания
<ALT>+	Просмотреть список точек прерывания (Breakpoints)
<ALT>+<K>	Просмотреть выполняемый тег (Stack)
<ALT>+<M>	Настроить логические псевдонимы путей к файлам (Mappings)
<ALT>+<Q>	Просмотреть список локальных переменных (Variables)
<ALT>+<R>	Просмотреть список набора данных (Recordsets)
<ALT>+<W>	Просмотреть значения выражений (Watch)
<ALT>+<X>	Переключить состояния точек прерывания
<ALT>+<Y>	Настроить отладку документа (Debug)
<CTRL>+<F4>	Редактировать текущий тег
<CTRL>+<F5>	Запустить/продолжить отладку текущего документа
<CTRL>+<F7>	Выделять подчеркиванием орфографические ошибки
<CTRL>+<F8>	Выполнить следующую строку кода (Step Into) при отладке документа

Таблица 3.3 (продолжение)

Клавиша или комбинация клавиш	Описание
<CTRL>+<F9>	Выполнить следующую строку кода, и если эта строка обращается к включенному файлу, то вызываемый модуль выполняется в фоновом режиме (Step Over) при отладке документа
<CTRL>+<F11>	Выполнить код до строки, на которой находится курсор или точка прерывания (Run to Cursor) при отладке документа
<CTRL>+<F12>	Переключиться в полноэкранный режим
<CTRL>+<,>	Вставить начало тега <>
<CTRL>+<.>	Вставить конец тега </>
<CTRL>+<3>	Добавить пару тегов для определения переменных (Surround with #)
<CTRL>+<A>	Выделить все
<CTRL>+	Жирный шрифт
<CTRL>+<C>	Копировать выделенный фрагмент
<CTRL>+<D>	Открыть в Macromedia Dreamweaver
<CTRL>+<E>	Открыть окно выбора тега
<CTRL>+<F>	Найти
<CTRL>+<G>	Перейти на строку, колонку
<CTRL>+<H>	Включить/выключить панель быстрого доступа к набору инструментов (QuickBar)
<CTRL>+<I>	Наклонный шрифт <i></i>
<CTRL>+<J>	Вызов окна со списком шаблонов кодов (Code Templates)
<CTRL>+<K>	Установить закладку
<CTRL>+<M>	Выделить тег, соответствующий текущему
<CTRL>+<N>	Создать новый файл
<CTRL>+<O>	Открыть файл
<CTRL>+<P>	Печать текущего документа
<CTRL>+<Q>	Повторить ввод последнего тега
<CTRL>+<R>	Заменить
<CTRL>+<S>	Сохранить текущий документ

Таблица 3.3 (продолжение)

Клавиша или комбинация клавиш	Описание
<CTRL>+<U>	Подчеркивание <u></u>
<CTRL>+<V>	Вставить из буфера обмена
<CTRL>+<W>	Заккрыть текущий документ
<CTRL>+<X>	Вырезать выделенный фрагмент
<CTRL>+<Y>	Удалить строку
<CTRL>+<Z>	Отменить последнее действие
<CTRL>+<ALT>+<F>	Запустить процесс автоматического формирования кода
<SHIFT>+<CTRL>+<F1>	Отразить результат расширенного поиска (Search) в окне вывода
<SHIFT>+<CTRL>+<F2>	Вывести список ошибок и предупреждений с комментариями (Validation) в окне вывода
<SHIFT>+<CTRL>+<F3>	Отразить список связей текущего документа (Links) в окне вывода
<SHIFT>+<CTRL>+<F4>	Показать все графические файлы из текущего каталога (Thumbnails) в окне вывода
<SHIFT>+<CTRL>+<F5>	Представить информацию относительно проектируемого файла (Deployment) в окне вывода
<SHIFT>+<CTRL>+<, >	Убрать отступ строки вправо
<SHIFT>+<CTRL>+<.>	Добавить отступ строки вправо
<SHIFT>+<CTRL>+<A>	Установить якорь
<SHIFT>+<CTRL>+	Конец строки
<SHIFT>+<CTRL>+<C>	Вставить код
<SHIFT>+<CTRL>+<E>	Вызвать построитель выражений
<SHIFT>+<CTRL>+<F>	Выполнить расширенный поиск
<SHIFT>+<CTRL>+<I>	Вставить рисунок
<SHIFT>+<CTRL>+<K>	Перейти к следующей закладке
<SHIFT>+<CTRL>+<L>	Включить/выключить окно вывода результатов тестирования (Results Window)
<SHIFT>+<CTRL>+<M>	Комментарий <!-- -->
<SHIFT>+<CTRL>+<P>	Добавить тег абзаца
<SHIFT>+<CTRL>+<R>	Выполнить расширенную замену

Таблица 3.3 (окончание)

Клавиша или комбинация клавиш	Описание
<SHIFT>+<CTRL>+<S>	Сохранить как
<SHIFT>+<CTRL>+<W>	Закрыть все документы
<SHIFT>+<CTRL>+<X>	Специальные символы
<SHIFT>+<CTRL>+<Z>	Вернуть последнее действие
<SHIFT>+<CTRL>+Пробел	Вставить неразрывный пробел
<SHIFT>+<F2>	Показать окно с перечнем доступных атрибутов текущей позиции
<SHIFT>+<F6>	Проверить текущий документ
<SHIFT>+<F7>	Проверить ошибки во всех документах
<SHIFT>+<F8>	Отобразить пользовательские настройки
<SHIFT>+<F9>	Переключиться между окном редактирования и таблицей ресурсов
<SHIFT>+<F10>	Показать контекстно-зависимое меню (щелчок правой кнопкой мыши)
<SHIFT>+<F12>	Сделать переход между режимами редактирования и помощи (для версии 5), или между режимами редактирования и проектирования (для предыдущих версий)

Для редактирования клавиатурных комбинаций быстрого доступа можно воспользоваться командой главного меню **Options | Customize** (Настройка | Настройки) или нажать комбинацию клавиш <Shift>+<F8>, а затем переключиться на вкладку **Keyboard Shortcuts** (Клавиши быстрого доступа).

Глава 4



ColdFusion Studio

В этой главе мы постараемся на примерах продемонстрировать работу с основными средствами в среде ColdFusion Studio. И первое, с чего мы начнем наше знакомство — это справочная система.

Справочная система

Каждый программный продукт в той или иной степени снабжен справочной системой. В одних программах эта система скудна и неудобна, в других — самодостаточна и удобна для использования. На взгляд автора система помощи ColdFusion Studio заслуживает отличную оценку.

Как было отмечено в *главе 3*, список разделов справочной системы располагается в панели **Resource** (Ресурс). После выбора вкладки **Help** (Справка)  нашему взору откроется справочное руководство, предоставленное разработчиками из Macromedia и Allaire, где можно получить полную информацию как о ColdFusion, так и о различных спецификациях, интересующих разработчиков Web-приложений (рис. 4.1).

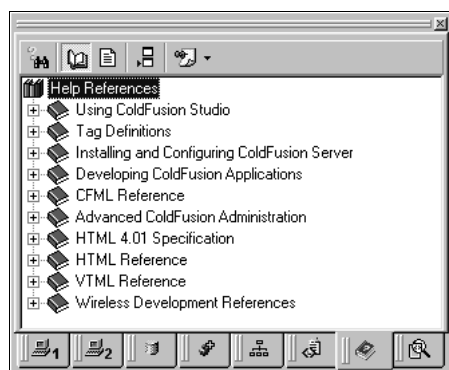


Рис. 4.1. Справочное руководство **Help References**

Активизировать вкладку **Help** (Справка) можно также, воспользовавшись главным меню программы **Help | Open Help References Window** (Справка | Открыть окно справочной системы).

Поиск

Обратите внимание на набор инструментов, расположенных вверху окна справочной системы, где первой по порядку стоит кнопка **Search** (Поиск) , позволяющая открывать диалоговое окно поиска (рис. 4.2).

Не закрывая диалоговое окно, можно наглядно познакомиться с правилами применения вспомогательных символов для построения сложных критериев поиска. Для этого достаточно нажать кнопку **Search Tips** (Поиск по ключу).

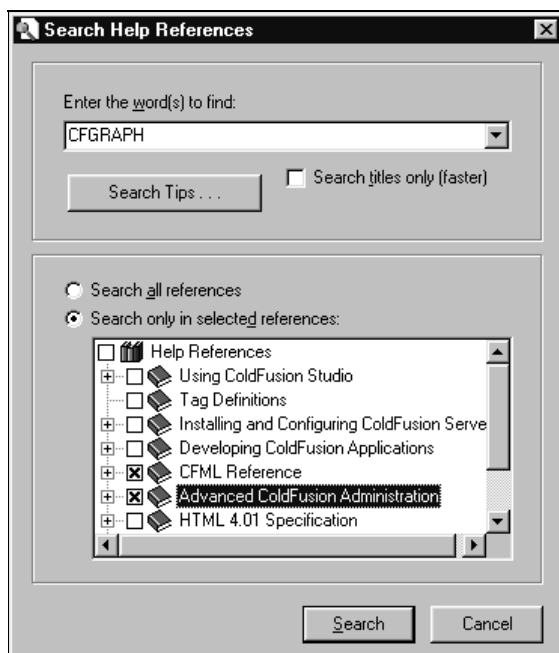


Рис. 4.2. Диалоговое окно поиска **Search Help References**

После ввода искомого фрагмента, указания интересующих разделов и подтверждения своего намерения в выполнении поиска на вкладке **Help** (Справка) отобразится список документов, удовлетворяющих вашим требованиям. Здесь стоит отметить, что вернуться к дереву справочного руководство можно с помощью второго по счету инструмента — **Help Reference Tree** (Дерево справочной системы)  — из того же верхнего набора инструментов.

Для того чтобы снова просмотреть список результатов поиска, необходимо воспользоваться инструментом **Search Results** (Результаты поиска) .

Далее, для получения детальной информации достаточно дважды щелкнуть мышью на одном из документов, входящих в список результата поиска. Отметим, что существует еще один инструмент — **Browse Help in Separate Panel**

(Показать справку в отдельной панели) , позволяющий регулировать расположение выбранного документа. При нажатой кнопке документ открывается в нижней части окна системы помощи (рис. 4.3). При другом состоянии данной кнопки документы с интересующей информацией отрываются в главном рабочем пространстве в режиме просмотра вспомогательной информации **Help** (Справка).

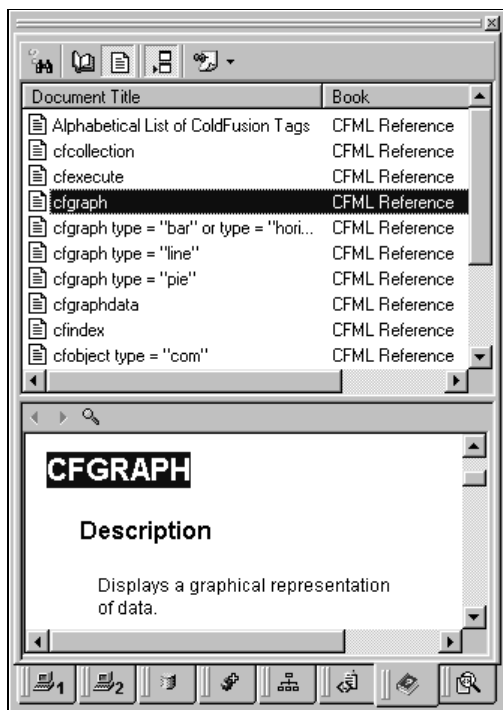


Рис. 4.3. Результат поиска

Пользовательские файлы помощи

ColdFusion Studio позволяет дополнять систему помощи собственными знаниями, в виде HTML-страниц и графических файлов. При этом графические файлы должны быть в формате GIF, JPG или PNG.

Для создания собственного раздела следует:

1. Открыть каталог **Help**, расположенный в корневом каталоге среды ColdFusion Studio (например, D:\Program Files\Macromedia\ColdFusion Studio 5\Help).
2. Создать собственный каталог, имя которого будет расцениваться как наименование раздела.

3. Скопировать или создать HTML-страницы, которые и станут неотъемлемой частью вашего раздела.
4. Обновить справочное руководство командой **Refresh** (Обновить) из контекстного меню или нажать клавишу <F5>.

На рис. 4.4 показан пример пользовательского раздела помощи.

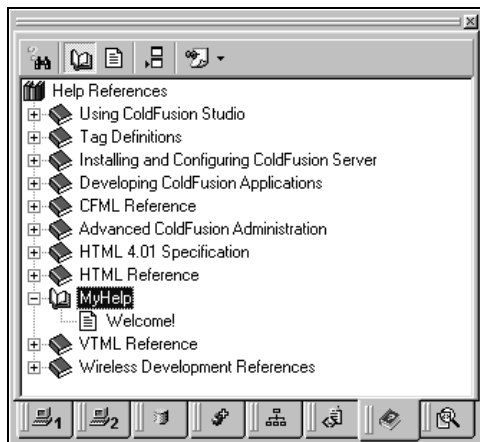


Рис. 4.4. Пользовательский раздел помощи

Обращаем ваше внимание на то, что список документов состоит не из имен файлов HTML-страниц, а из заголовков этих самых HTML-страниц. То есть текста, заключенного между тегами <TITLE></TITLE> в коде страниц.

Закладки

Последний инструмент, о котором еще не было сказано ни слова, — это закладки **Bookmarks** . С его помощью можно формировать список собственных закладок на часто используемые страницы. Все достаточно просто: щелкаете мышью по инструменту, открывается окно просмотра закладок. После чего для добавления новой закладки нажимаете кнопку **Add** (Добавить) и заполняете необходимые поля (рис. 4.5).

Но существует и более простой способ: щелкаете правой кнопкой мыши на документе, требующем повторного обращения, и из контекстного меню выбираете команду **Add bookmark** (Добавить закладку).

В дальнейшем для активизации необходимого документа, на который была сделана закладка, достаточно раскрыть список закладок, нажав на прилегающую к инструменту стрелку (рис. 4.6).

Обратите внимание на инструменты, относящиеся к нижнему окну: это две стрелки, направленные в противоположные стороны, и инструмент, напоминающий лупу. Первые два инструмента — это ничто иное, как средство пере-

хода по посещаемым страницам: **Previous** (Предыдущий) и **Next** (Следующий). А так называемая лупа **Open in External Browser** (Открыть во внешнем браузере) позволяет открывать страницу в отдельном окне браузера.

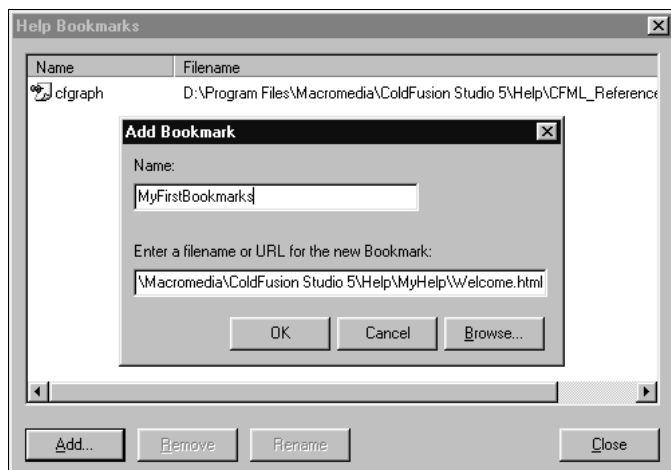


Рис. 4.5. Диалоговое окно добавления закладки **Add Bookmark**



Рис. 4.6. Пример использования закладок

Для удаления или редактирования закладок необходимо из раскрывающегося списка закладок, появляющегося при нажатии кнопки-стрелки справа от инструмента **Bookmarks** (Закладки), выбрать пункт **Organize Bookmarks** (Организовать закладки) или, щелкнув мышью непосредственно по инструменту **Bookmarks** (Закладки), в открывшемся диалоговом окне **Help Bookmarks** (Справка о закладках) воспользоваться кнопками **Remove** (Удалить) или **Rename** (Переименовать) в зависимости от поставленной задачи.

Контекстная помощь

Очень часто в разных приложениях встречается контекстная помощь, когда при выделении одной из команд, функций или одного из тегов при нажатии функциональной клавиши <F1> открывается документ, содержащий вспомогательную информацию, именно для этой команды, функции или тега.

ColdFusion Studio — не исключение: для получения сведений о текущем теге не обязательно его выделять, на него достаточно установить текстовый курсор и нажать клавишу <F1> или воспользоваться главным меню программы **Help | Show Help for Current Tag** (Справка | Справка о текущем теге).

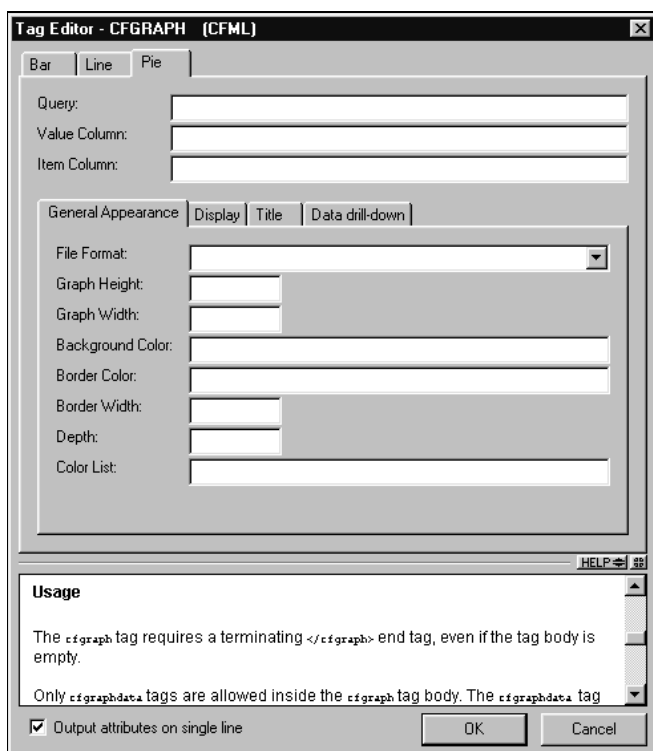


Рис. 4.7. Пример диалогового окна с текстом справки

Кнопки помощи в диалоговых окнах

В *главе 3* мы уже говорили о кнопках помощи в диалоговых окнах. Однако стоит еще раз напомнить, что каждое диалоговое окно редактирования тегов включает кнопки помощи.

Одна кнопка с надписью **Help** (Справка)  позволяет в нижней части диалогового окна показать вспомогательное описание.

На рис. 4.7 приведен пример диалогового окна, содержащего текст справки. Другая кнопка, напоминающая перекрестие , позволяет отобразить текст справки с информацией о текущем инструменте в новом окне.

Управление файлами и каталогами

Благодаря встроенной системе управления файлами и каталогами, ColdFusion Studio имеет еще одно преимущество в копилке своих достоинств.

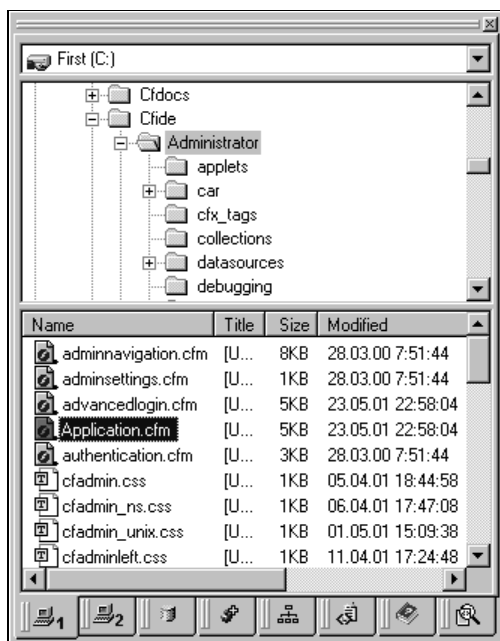


Рис. 4.8. Окно управления файлами и каталогами

Система управления файлами и каталогами обеспечивает доступ ко всем файлам, включая сетевые ресурсы и удаленные соединения. Здесь вы можете добавлять, изменять и удалять каталоги и файлы, фильтровать просмотр по типам файлов, предварительно просматривать графические файлы, соз-

давать список избранных каталогов для быстрого доступа. Но обо всем по порядку.

Начиная с пятой версии, для облегчения работы с файловой системой на панели **Resource** (Ресурс) представлены две вкладки для работы с файлами



Выбрав вкладку **Files 1** (Файлы 1) или **Files 2** (Файлы 2), вы увидите окно, разбитое на три условных области:

- ☐ раскрывающийся список **Drive List** (Список устройств);
- ☐ список каталогов в виде дерева **Directory List** (Список каталогов);
- ☐ список файлов **File List** (Список файлов).

На рис. 4.8 показано окно управления файлами и каталогами для вкладки **Files 1** (Файлы 1).

Высоту списков каталогов и файлов можно регулировать, используя горизонтальную линию разбиения между этими списками.

Работа с каталогами

Для выполнения каких-либо действий с каталогами необходимо щелкнуть на этом каталоге правой кнопкой мыши и из контекстного меню выбрать необходимую команду.

Предоставленный список команд позволяет копировать, перемещать, удалять и переименовывать каталоги. А также с помощью этого списка можно выполнять такие действия, как просмотр свойств, предоставление доступа к каталогам, открытие в отдельном окне и организация поиска.

Для создания нового каталога необходимо:

1. В списке каталогов выбрать тот, в котором будет создан будущий каталог.
2. Щелкнуть правой кнопкой мыши в списке файлов.
3. В контекстном меню выбрать команду **Create Folder** (Создать папку).
4. Отредактировать название нового каталога.

Выделяя тот или иной каталог, в нижнем окне вы отображаете список файлов, находящихся в выбранном каталоге.

Работа с файлами

Для выполнения каких-либо действий с файлами так же, как и с каталогами, необходимо щелкнуть на файле правой кнопкой мыши и из контекстного меню выбрать необходимую команду.

По умолчанию в списке файлов перечисляются все Web-файлы. Однако, используя команду фильтрации (**Filter** (Фильтр)), можно изменять режим

вывода файлов, отражая только Web-документы, только Web-рисунки или все файлы.

С помощью команды просмотра (**View** (Вид)) можно включать/отключать просмотр данных о файлах (заголовок, размер, дата и время последнего изменения, тип).

Также в предоставленном списке команд можно копировать, перемещать, удалять и переименовывать выбранные файлы.

HTML-страницы, как и страницы CFML, ASP, PHP, открываются в рабочей области для редактирования после двойного щелчка мышью на этом файле. Двойной щелчок мышью по графическому файлу приводит к вызову диалогового окна создания ссылки на выбранный рисунок. Впрочем, графический файл достаточно перетащить указателем мыши в рабочую область, чтобы в текущем документе сформировался код с подобной ссылкой.

Работа со списком избранных каталогов

По мере разработки Web-приложений количество этих приложений растет. В свою очередь, каждое Web-приложение имеет собственный путь размещения, и на определенном этапе уже не так просто ориентироваться в собственных разработках. А между тем, при создании очередного проекта очень даже удобно обратиться к уже созданному Web-приложению для просмотра готовых решений.

В среде ColdFusion Studio предусмотрена возможность создания списка избранных каталогов, что позволяет с легкостью перемещаться по существующим приложениям.

Для добавления текущего каталога в список избранных достаточно, находясь на одной из вкладок **Files 1** (Файлы 1) или **Files 2** (Файлы 2), щелкнуть правой кнопкой мыши на списке файлов и из контекстного меню выбрать команду **Favorite Folders | Add Current Folder to Favorites** (Избранные папки | Добавить текущую папку к избранным).

Для управления списком избранных каталогов необходимо из того же контекстного меню выбрать команду **Favorite Folders | Organize Favorites** (Избранные папки | Организовать избранные), где вы можете добавлять и удалять избранные каталоги, используя кнопки **Add** (Добавить) и **Remove** (Удалить), а также изменять очередность каталогов с помощью кнопок-стрелок, расположенных в диалоговом окне **Favorite Folders** (Избранные папки). На рис. 4.9 показано диалоговое окно управления списком избранных каталогов.

По мере пополнения списка каталогов контекстное меню пункта **Favorite Folders** (Избранные папки) увеличивается, и для перехода к очередному Web-приложению, расположенному в одном из добавленных каталогов, достаточно воспользоваться предоставленным меню (рис. 4.10).

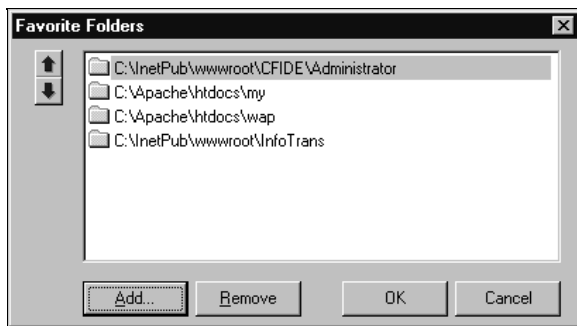


Рис. 4.9. Диалоговое окно **Favorite Folders**

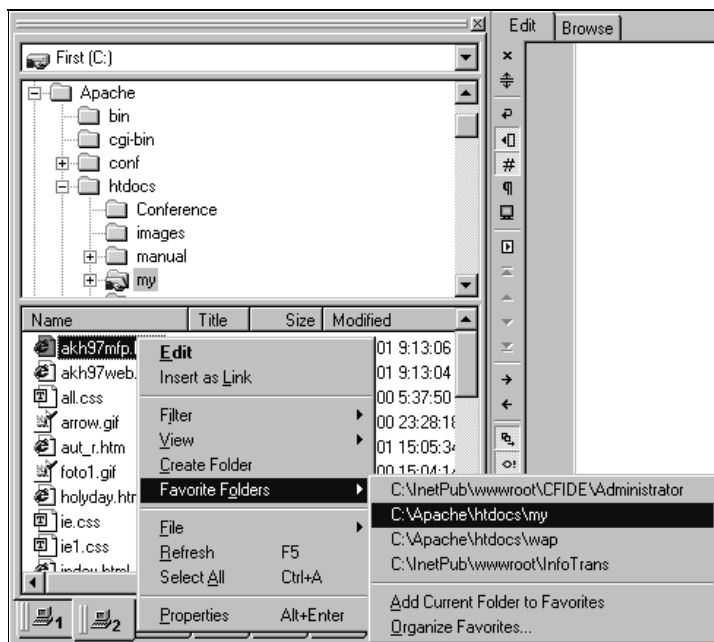


Рис. 4.10. Контекстное меню списка избранных каталогов

Еще раз отметим, что использование избранных каталогов полезно в случае работы над несколькими проектами, когда вам достаточно часто приходится обращаться к разным каталогам.

Работа с удаленными серверами

В разд. "Установка соединения с удаленным сервером" гл. 2 мы уже рассказывали о встроенной в ColdFusion Studio возможности удаленного соединения с помощью RDS- и FTP-клиентов.

Обратите внимание на содержание списка устройств, где одним из пунктов является **Macromedia FTP & RDS** (рис. 4.11).

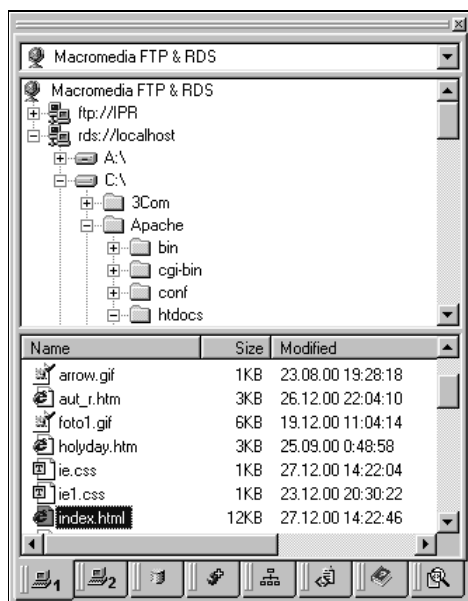


Рис. 4.11. Пункт **Macromedia FTP & RDS**

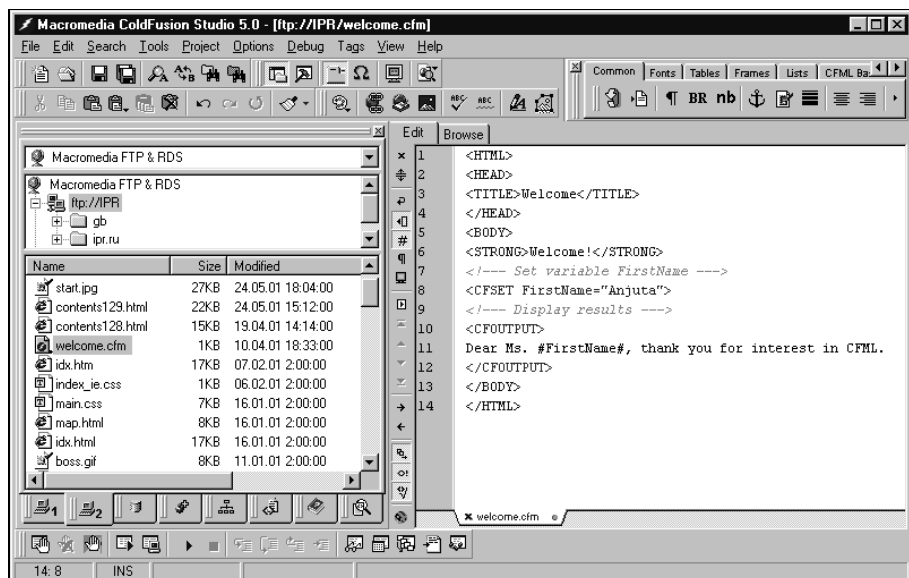


Рис. 4.12. Работа с удаленным документом

Здесь же, на вкладках управления файлами **Files 1** (Файлы 1) и **Files 2** (Файлы 2), не выходя из среды, можно добавлять и изменять соединения как к FTP-, так и к RDS-серверам.

Например, установив новое соединение с FTP-сервером (в нашем случае IPR) и сделав это соединение активным с помощью двойного щелчка мыши, в списке файлов мы обнаружим содержимое FTP-сервера в виде перечня документов. Такой список позволит работать с данными документами непосредственно в среде ColdFusion Studio (рис. 4.12).

Обратите внимание, если вы открываете документ, используя удаленное соединение, то это отражается на вкладке данного документа, помечая его зеленым индикатором.

Загрузка Web-страниц

Вы можете загружать любую HTML-страницу в редакторе ColdFusion Studio. Эти файлы не могут быть обратно сохранены на сервере.

Загрузка или, скажем иначе, открытие страницы производится в следующем порядке:

1. Выберите из главного меню команду **File | Open from Web** (Файл | Открыть из Web).
2. Введите URL страницы или используйте кнопки **Favorites** (Избранные) в Internet Explorer или **Bookmarks** (Закладки) в Netscape Navigator в зависимости от ситуации.
3. При необходимости организовать доступ через Proxy-сервер нажмите кнопку **Proxy** (Прокси) и введите имя Proxy и номер порта.
4. Вы можете произвольно устанавливать предел прерывания связи, изменяя значение **Timeout after** (Ожидание).
5. Подтвердите открытие страницы нажатием кнопки **OK**.

На рис. 4.13 показано диалоговое окно открытия Web-страницы.

Заметим, что открывающиеся страницы будут поименованы как Untitled1, Untitled2, Untitled3 и т. д.

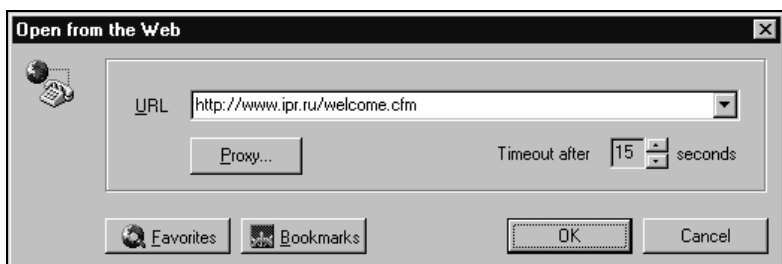


Рис. 4.13. Диалоговое окно открытия Web-страницы **Open from the Web**

Работа с проектами

Проектирование сложных Web-приложений подразумевает включение большого числа файлов различных типов, от обычных HTML-страниц, стилевых таблиц и графических файлов до CFML-страниц и JavaScript-файлов. Облегчить решение этой задачи в среде ColdFusion Studio призван модуль проектов **Projects** (Проекты).

Окно управления проектами располагается в панели **Resource** (Ресурс). После выбора вкладки **Projects** (Проекты)  открывается окно, очень похожее на окно файлов за исключением того, что вместо выбора устройств раскрывающийся список служит для указания текущего проекта, а вместо списка привычных каталогов отображается перечень виртуальных и физических каталогов, а также список ресурсов.

Проект в нашем случае подразумевает совокупность файлов, входящих в Web-проект. При создании Web-приложения необязательно создавать проект и использовать встроенные функции в данном контексте, однако создание проекта позволит:

- ☐ размещать все необходимые файлы для создания Web-приложения в легкодоступном месте в рамках проекта ColdFusion Studio;
- ☐ после завершения разработки приложения или, назовем иначе, сайта загружать весь проект на Web-сервер и быть уверенным, что ни один файл не будет потерян;
- ☐ осуществлять обслуживающие операции, такие как поиск, перемещение и проверку ссылок целиком в проекте, а не в каждом отдельном файле.

Создание нового проекта

В перечне инструментов, расположенных над окном проектов, находится кнопка **New Project** (Новый проект) , которая, позволяя создавать новый проект, открывает диалоговое окно создания проекта (рис. 4.14).

При создании проекта достаточно указать название и путь размещения нового проекта, как тут же после нажатия кнопки **ОК** в списке проектов появится новая запись, а списки каталогов и файлов будут обновлены в соответствии с содержанием проекта.

Обратите внимание, что в диалоговом окне создания проектов можно сбросить флажок **Add all subfolders, and include files of the following types** (Добавлять все подкаталоги и файлы следующих типов), что позволит не включать в новый проект подкаталоги и файлы, располагающиеся по указанному пути размещения проекта. Можно в раскрывающемся списке **File Types (separate extantions with semi-colons)** (Типы файлов (через точку с запятой)) указать выборочное подключение файлов к новому проекту в зависимости от типа файлов.

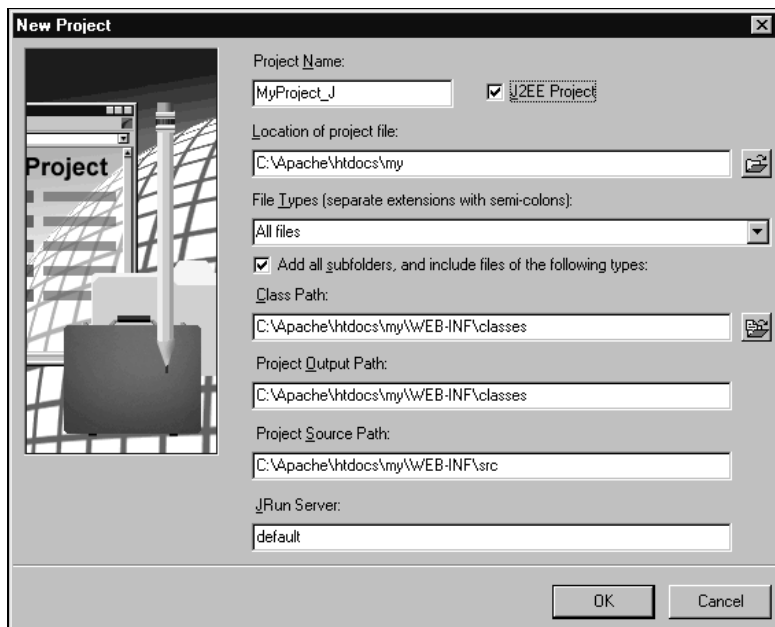


Рис. 4.14. Диалоговое окно создание проекта **New Project**

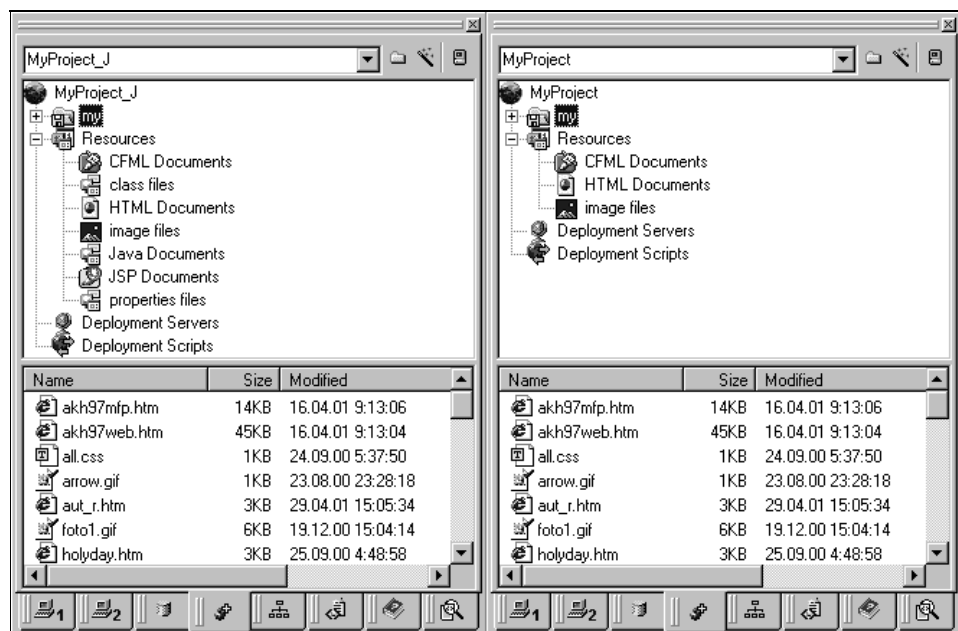


Рис. 4.15. Сравнение двух проектов с включение J2EE (слева) и без включения данной технологии (справа)

Флажок **J2EE Project** (Проект J2EE) позволяет включить дополнительные параметры, предназначенные для реализации проекта с использованием языка Java. Рис. 4.15 демонстрирует два проекта с включением J2EE (Java 2 platform Enterprise Edition) и без включения данной технологии.

Для справки, JRun Server — это один из программных продуктов в прошлом компании Allaire, в настоящий момент принадлежащий так же, как и ColdFusion, корпорации Macromedia. Он является инструментом для создания и развертывания серверных приложений, использующих Java-сервлеты и Java Server Pages (JSP).

Добавим, что файл-проект среды ColdFusion Studio имеет расширение `apf`.

Каталоги проекта

Каталоги служат основным средством организации файлов в проекте. В ColdFusion Studio существует два вида каталогов:

- ☐ **Physical Folders** — физические каталоги;
- ☐ **Virtual Folders** — виртуальные каталоги.

Физические каталоги относятся к определенной папке вашего компьютера.

Виртуальные каталоги служат репозиториями для файлов, не имеющих логической связи между собой. Вы можете использовать виртуальный каталог как некий контейнер, куда можно помещать любые файлы. Виртуальные каталоги позволяют организовать в одном месте ColdFusion Studio все разбросанные файлы.

Физические каталоги в свою очередь можно разделить еще на два вида:

- ☐ **Manual-inclusive** — каталоги ручного вложения;
- ☐ **Auto-inclusive** — каталоги автоматического вложения.

Каталог ручного вложения требует, чтобы вы определили, какие файлы в отображенной папке необходимо вложить в каталог. Например, в вашем каталоге находится 7 файлов, и только два из них необходимы для использования в Web-приложении. Вы можете создать каталог ручного вложения, поместив туда оба файла.

Каталог автоматического вложения включает каждый файл в отображенной папке по определению. Вы можете определить только тот формат файлов, который хотите включить (например, все графические файлы). Каталоги данного вида обновляются автоматически при добавлении или удалении файлов из данной папки.

При добавлении каталога к проекту вам необходимо решить, куда поместить данный каталог. Вы можете добавить каталог к корню проекта или к уже существующему каталогу. При добавлении к существующему каталогу новый каталог обозначается как *дочерний*, а существующий каталог — как *ро-*

дительский. Существуют некоторые ограничения, связанные с добавлением виртуальных каталогов: виртуальные каталоги могут быть добавлены только к корню проекта или к другому виртуальному каталогу. Вы не можете добавлять виртуальные каталоги к каталогам ручного или автоматического вложения, однако имеете право добавлять каталоги ручного и автоматического вложения к виртуальным каталогам.

Для добавления виртуального каталога следует:

1. Щелкнуть правой кнопкой мыши на родительском каталоге: на корневой папке проекта или на другом виртуальном каталоге.
2. Выбрать из контекстного меню команду **Add Folder** (Добавить папку). Перед вами появится диалоговое окно добавления каталога (рис. 4.16).
3. Ввести название каталога в поле **Folder Name** (Имя папки).
4. Выбрать переключатель **Virtual Folder** (Виртуальная папка) и подтвердить нажатием кнопки **OK**.

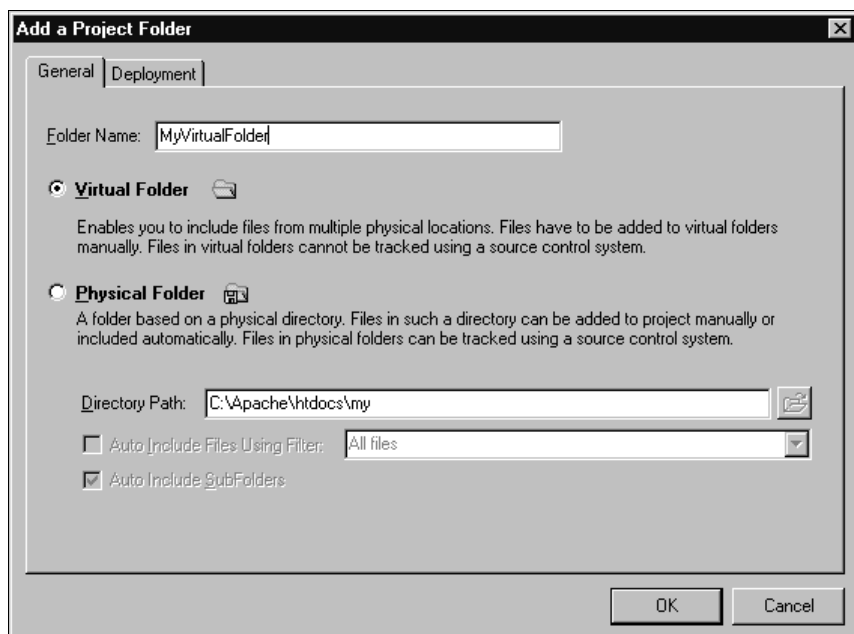


Рис. 4.16. Диалоговое окно добавления каталога **Add a Project Folder**

Добавление остальных типов каталогов осуществляется аналогичным образом — указанием необходимых параметров. На рис. 4.17 приведен результат добавления разных типов каталогов.

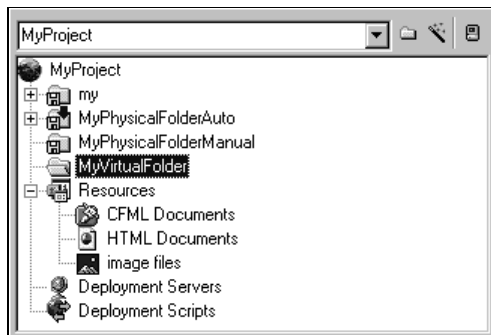


Рис. 4.17. Варианты каталогов

Для изменения свойств каталога достаточно щелкнуть правой кнопкой мыши по необходимому каталогу и выбрать из контекстного меню команду **Properties** (Свойства), что позволит редактировать параметры каталога с помощью дополнительного диалогового окна.

Итак, еще раз перечислим возможные каталоги (табл. 4.1).

Таблица 4.1. Каталоги

Значок	Название	Описание
	Виртуальный каталог	Используйте, когда хотите организовать файлы из разных мест в одном пространстве ColdFusion Studio
	Физический каталог автоматического включения	Используйте, если вы желаете, чтобы все файлы или файлы определенного формата были объединены в текущем проекте в специальном каталоге
	Физический каталог ручного вложения	Используйте, если вы желаете, чтобы избранные файлы были включены в текущий проект и объединены в специальном каталоге

На любой каталог распространяется очень удобное правило открытия сразу всех документов, находящихся в одном каталоге: щелкните правой кнопкой мыши по значку каталога и выберите из появившегося контекстного меню команду **Open All Documents In Folder** (Открыть все документы из папки).

Файлы проекта

Теперь, когда у вас уже есть представление о том, как манипулировать каталогами проекта, вы можете приступить к работе с файлами. Методы работы с файлами могут различаться в зависимости от типа каталога, содержащего данный файл.

Файлы каталогов ручного вложения и виртуальных каталогов

Процедура добавления и удаления файлов одинакова для каталогов ручного вложения и виртуальных каталогов.

Чтобы выполнить добавление файлов в каталог ручного вложения и виртуальный каталог:

1. Щелкните правой кнопкой мыши по значку каталога и укажите команду **Add Files to Folder** (Добавить файлы в папку). Появится диалоговое окно выбора файлов.
2. Выберите файлы, которые вы хотите добавить в каталог проекта. Для выбора более чем одного файла используйте клавиши <Shift> и <Ctrl>.
3. Подтвердите добавление файлов нажатием кнопки **Add** (Добавить).

Удаление файлов из каталога ручного вложения и виртуального каталога выполняется следующим образом:

1. Выделите каталог. Все файлы, содержащиеся в выбранном каталоге, отображаются в нижнем списке **File List** (Список файлов).
2. Щелкните правой кнопкой мыши по значку файла, который вы хотите удалить, и выберите команду **Remove from this folder** (Удалить из этой папки).
3. Подтвердите удаление файла положительным ответом на поставленный вопрос.

Файлы в каталогах автоматического вложения

Вы можете контролировать файлы, хранящиеся в каталоге автоматического вложения, определяя их типы, а также можете изменить типы файлов.

Чтобы изменить тип файлов в каталогах автоматического вложения:

1. Щелкните правой кнопкой мыши по значку каталога и выберите команду **Properties** (Свойства).
2. Модифицируйте расширение для тех типов файлов, которые вы хотите вложить в каталог. Можете изменить расширение с помощью раскрывающегося списка или вручную.
3. Подтвердите изменения типов файлов нажатием кнопки **ОК**.

Ресурсы проекта

Структура ресурсов позволяет видеть файлы проекта, распределенные по типам. Например, щелкнув на строке **CFML Documents** (CFML-документы), вы можете увидеть список всех CFML-файлов вашего проекта.

Структура ресурсов состоит из семи категорий — **CFML Documents** (CFML-документы), **class files** (class-файлы), **HTML Documents** (HTML-документы), **image files** (изображения), **Java Documents** (Java-документы), **JSP Documents** (JSP-документы) и **properties files** (файлы свойств) — для проектов с использованием технологии J2EE и из трех типов ресурсов — **CFML Documents** (CFML-документы), **HTML Documents** (HTML-документы) и **image files** (изображения) — для проектов без использования J2EE. Вы также можете создать собственные ресурсы проекта и определять типы файлов для этих ресурсов.

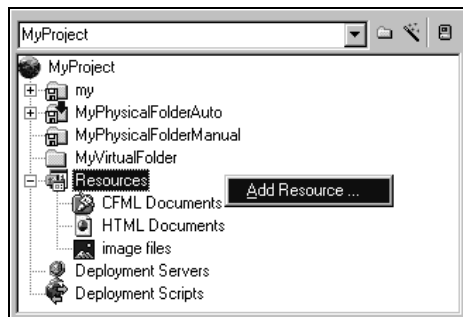


Рис. 4.18. Раздел **Resources**

Добавление ресурса выполняется следующим образом:

1. Щелкните правой кнопкой мыши на пункте **Resources** (Ресурсы) и выберите из контекстного меню команду **Add Resource** (Добавить ресурс) — рис. 4.18. Появится диалоговое окно **Add Resource Folder** (Добавить папку ресурса) — рис. 4.19.
2. Введите название ресурса в текстовом поле **Resource Name** (Имя ресурса).
3. Введите расширения файлов для данного типа ресурса в поле **Resource Filter** (Фильтр ресурса), разделяя их с помощью точки с запятой, если их более одного (например, для фильтра ресурса документов JavaScript — js).
4. Подтвердите добавление нового ресурса нажатием кнопки **ОК**.

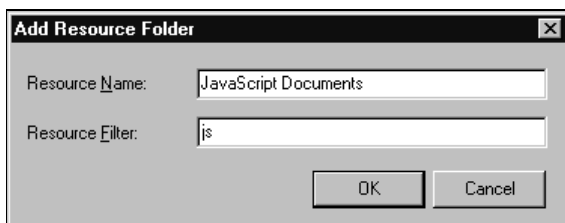


Рис. 4.19. Диалоговое окно добавления ресурса **Add Resource Folder**

После щелчка по значку добавленного ресурса в списке файлов появятся документы, входящие в текущий проект и имеющие указанные расширения (рис. 4.20).

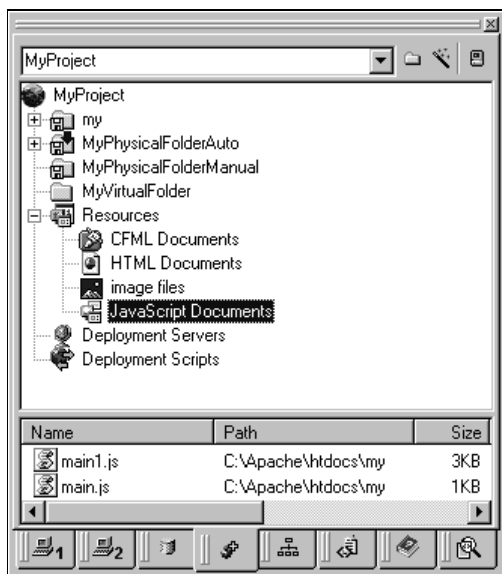


Рис. 4.20. Новый ресурс **JavaScript Documents**

Чтобы изменить свойства ресурса:

1. Щелкните правой кнопкой мыши на пункте **Resources** (Ресурсы) и выберите команду **Properties** (Свойства). Появится диалоговое окно **Edit Resource Folder** (Изменить папку ресурса).
2. Сделайте необходимые изменения названия или фильтров ресурса и нажмите кнопку **OK**.

Для удаления ресурса:

1. Щелкните правой кнопкой мыши на пункте **Resources** (Ресурсы) и выберите команду **Remove Resource** (Удалить ресурс).
2. Подтвердите удаление ресурса положительным ответом на поставленный вопрос.

Добавление, правку и удаление ресурсов можно осуществить в настройках среды ColdFusion Studio, выбрав пункт меню **Options | Settings** (Настройка | Параметры) или нажав клавишу <F8>. В появившемся диалоговом окне **Settings** (Параметры) нужно активизировать раздел **Projects** (Проекты), где и следует внести необходимые изменения.

Использование инспектора тегов

Окно, предназначенное для работы с тегами, располагается в панели **Resource** (Ресурс) и обозначается вкладкой **Tag Inspector** (Инспектор тегов) . Данная вкладка содержит два основных элемента со вспомогательными инструментами. Этими элементами являются расположенное в верхней части окна дерево тегов **Tag Tree** и размещенный в нижней части инспектор тегов **Tag Inspector** (рис. 4.21). С помощью этих элементов вам будут доступны только CFML- и HTML-теги, что упрощает создание и редактирование тегов.

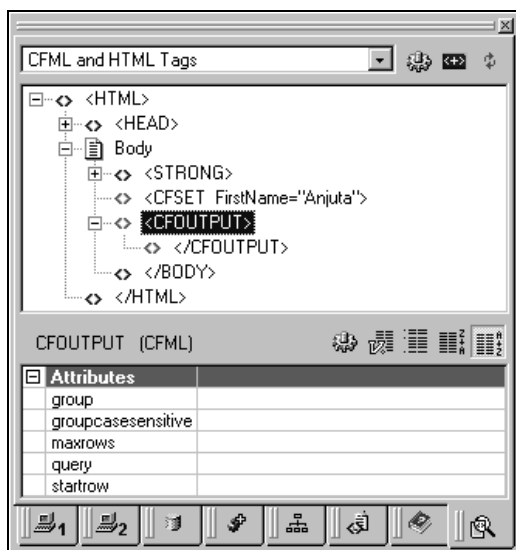


Рис. 4.21. Окно для работы с тегами **Tag Inspector**

Выбор некоторого документа приводит к обновлению содержимого данного окна в соответствии со структурой кодов этого документа.

Дерево тегов

Дерево тегов позволяет легко перемещаться и проверять структуру кода документа. Также с его помощью можно отображать только те теги, которые вас интересуют.

Для установки фильтрации отображения тегов используйте раскрывающийся список, находящийся в верхней части окна. Более того, список фильтров можно пополнять и редактировать, для этих целей используйте ближайший

инструмент, похожий на шестеренку — **Configure Outline Profiles** (Настроить профили вывода) .

Инструмент в виде плюса **Collapse document based on outline** (Свернуть документ в соответствии со структурой)  позволяет сворачивать тексты в рабочей области редактирования документа в соответствии с композицией дерева тегов. То есть, если в области дерева тегов какие-либо теги раскрыты не полностью (отмечены плюсом), то и по тексту документа данные теги будут "свернуты".

В разд. "Настройку ColdFusion Studio" гл. 2 мы уже говорили о сворачиваемых текстах.

Хотелось бы отметить, что сворачивать тексты можно не только в соответствии с иерархией тегов, но и произвольно, выделив текст в режиме редактирования и нажав одну из кнопок в виде минуса, находящихся в поле для отражения нумерации строк, переносов, закладок и отслеживания точек прерывания (рис. 4.22).

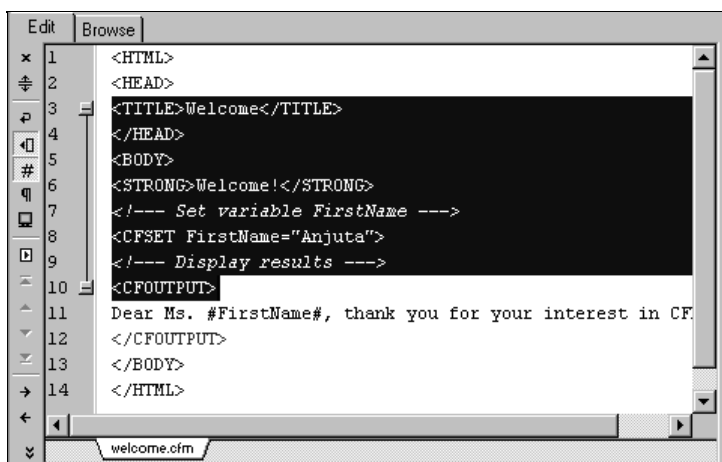


Рис. 4.22. Выделение текста для сворачивания

На рис. 4.23 показана область редактирования документа после выполнения команды сворачивания выделенного текста.

Для раскрытия свернутого текста достаточно дважды щелкнуть по маркеру, показывающему первые несколько символов свернутой выборки.

Вернемся к дереву тегов. Для обновления дерева тегов после правки документа используйте инструмент **Refresh** (Обновить) .

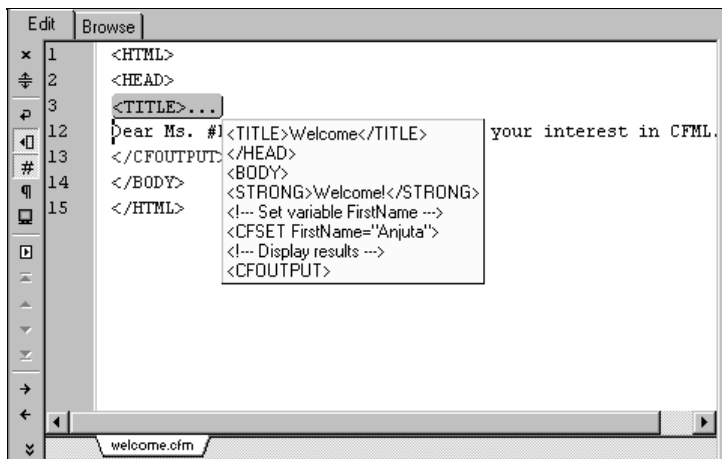


Рис. 4.23. Результат сворачивания текста

Инспектор тегов

Что касается инспектора тегов, этот элемент по своему образу и подобию очень похож на средство, присутствующее во многих объектно-ориентированных приложениях. Выбирая тот или иной объект, в нашем случае — тег, все его свойства выстраиваются в некую таблицу с отображением значений и возможностью визуального изменения этих значений.

Для редактирования свойств тега необходимо:

1. Выделить тег в области дерева тегов.
2. Выбрать необходимое для редактирования свойство в области инспектора тегов.
3. Ввести значение в соответствии с требованием относительно данного свойства.

На рис. 4.24 приведен пример редактирования свойств тегов с применением инспектора тегов.

Для удобства список свойств можно сортировать. Существуют следующие варианты сортировки:

- ☐ относительно версий (**Version Specific**);
- ☐ по категориям (**Categorized**);
- ☐ по алфавиту в обратном порядке (**Alphabetically Z-to-A**);
- ☐ по алфавиту (**Alphabetically A-to-Z**), установленный по умолчанию.

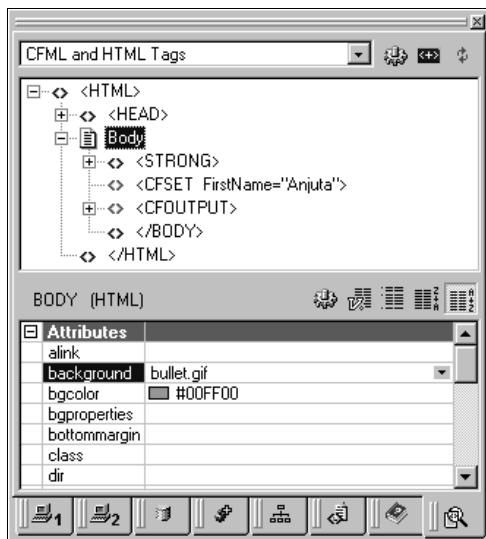


Рис. 4.24. Редактирование свойств тега

Обозреватель сайта

Окно обозревателя сайта располагается в панели **Resource** (Ресурс) в качестве вкладки **Site View** (Обозреватель сайта)  (рис. 4.25). Обозреватель сайта обеспечивает наглядное отображение всех связей текущего документа с другими файлами.

Данное средство облегчает работу со связями: достаточно щелкнуть мышью по одной из ссылок, чтобы активизировать код этой связи в рабочей области редактирования документа. При двойном щелчке мыши по интересующей ссылке раскрывается "ветка" связей того документа, с которым связана выбранная ссылка. К примеру, обратите внимание на рис. 4.26, где мы попытались отразить связи ссылки **Sample**. При повторном двойном щелчке мыши по ссылке "ветки" связей будут скрыты.

Как вы можете убедиться, при большом количестве связей не очень удобно, а, по сути, совсем неудобно просматривать список данных связей. Решить эту проблему можно, изменив стиль просмотра. Для этого достаточно щелкнуть правой кнопкой мыши в области окна обозревателя сайта и выбрать команду **View Style | Tree** (Стиль обозревателя | Дерево). На рис. 4.27 показано окно обозревателя сайта в стиле дерева.

Для редактирования связанного документа необходимо:

1. Выбрать ссылку.
2. Щелкнуть правой кнопкой мыши и из контекстного меню выбрать команду **Edit** (Правка).

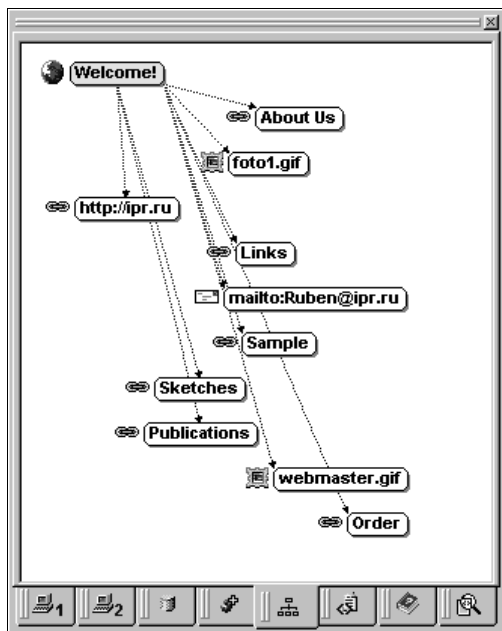
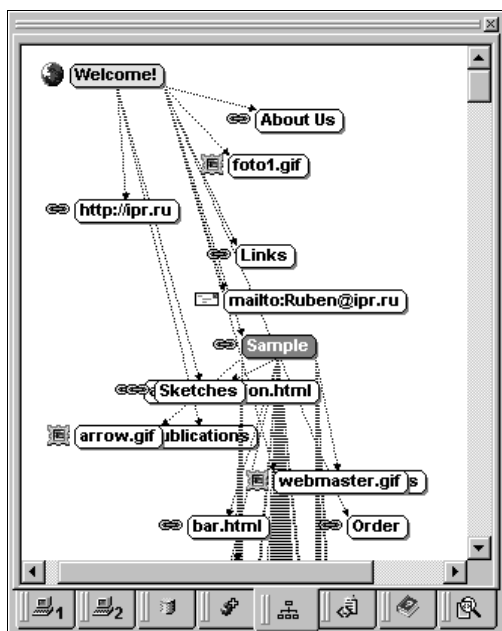
Рис. 4.25. Окно обозревателя сайта **Site View**

Рис. 4.26. Раскрытие связей в окне обозревателя сайта

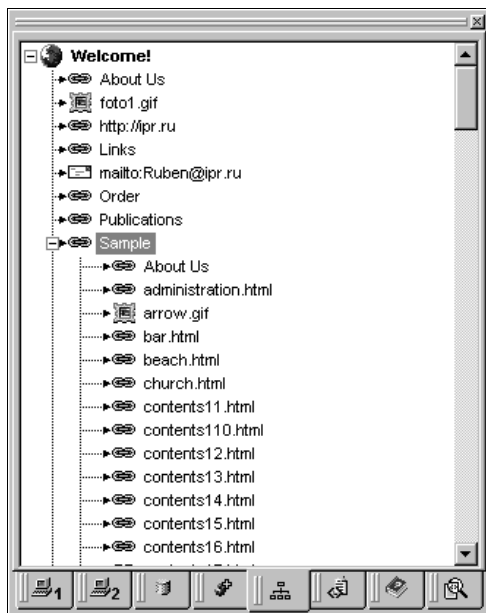


Рис. 4.27. Окно обозревателя сайта в стиле дерева

В этом случае область обозревателя сайта обновится в соответствии с выбранным для редактирования документом.

Многократно повторяющиеся фрагменты кода

Окно для работы с фрагментами кода располагается в панели **Resource** (Ресурс) в качестве вкладки **Snippets** (Фрагменты кода) . Данный модуль обеспечивает работу с пользовательскими многократно повторяющимися фрагментами кода.

Предположим, необходимо сформировать собственный фрагмент кода для его многократного добавления в наши документы.

1. Итак, выбрав вкладку **Snippets** (Фрагменты кода), скорее всего вы обнаружите "чистый белый лист". Вот с него и начнем.
2. Щелкаем правой кнопкой мыши и из "скромного" списка выбираем команду создания каталога **Create Folder** (Создать папку).
3. Вводим название каталога, например `MyFolderSnippets`.
4. Затем нажимаем правую кнопку мыши уже на созданном нами каталоге и выбираем команду добавления фрагмента кода **Add Snippet** (Добавить фрагмент кода).

5. После чего заполняем диалоговое окно **Snippet** (Фрагмент кода) — рис. 4.28, и подтверждаем создание нашего "лоскутка" нажатием кнопки **ОК**.

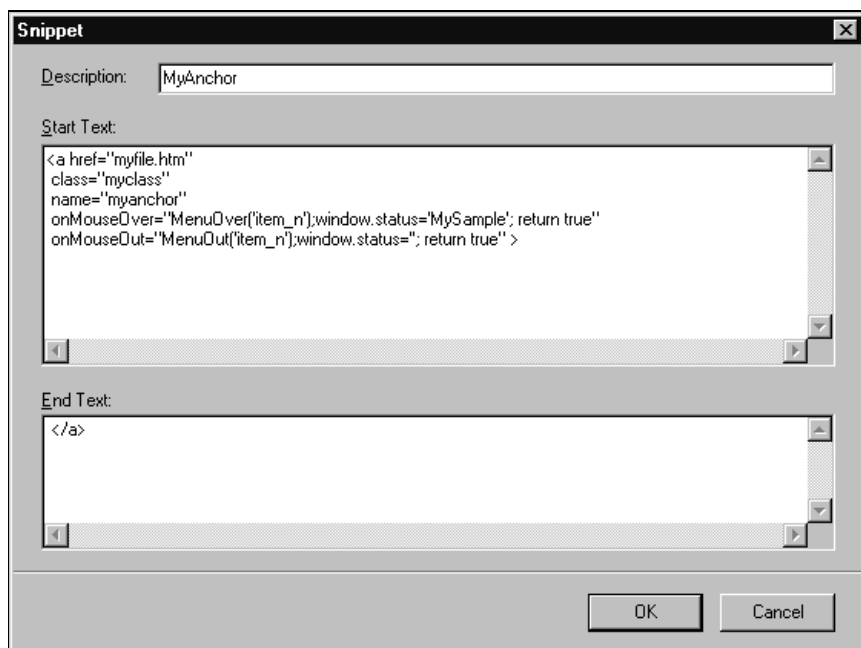


Рис. 4.28. Диалоговое окно добавления фрагмента кода **Snippet**

Диалоговое окно для формирования фрагмента кода имеет поле ввода **Description** (Описание), куда обязательно следует внести название кода, и две текстовые области — **Start Text** (Начальный текст) и **End Text** (Конечный текст) — непосредственно для ввода содержимого кода. Начальный текст выступает в качестве текста, помещаемого до текстового курсора, а конечный текст — соответственно после. Обе текстовые области необязательны. Только есть ли смысл создавать фрагмент кода без этого самого кода?

Теперь, когда у нас уже есть собственный фрагмент кода, в чем можно убедиться, глядя на рис. 4.29, необходимо научиться им пользоваться.

Для внедрения данного фрагмента в текущий документ достаточно просто дважды щелкнуть по данному фрагменту. На рис. 4.30 представлен результат такого внедрения.

Если же вы желаете отредактировать фрагмент кода, воспользуйтесь правой кнопкой мыши и выполните команду редактирования **Edit Snippet** (Правка фрагмента кода).



Рис. 4.29. Окно для работы с фрагментами кодов **Snippets**

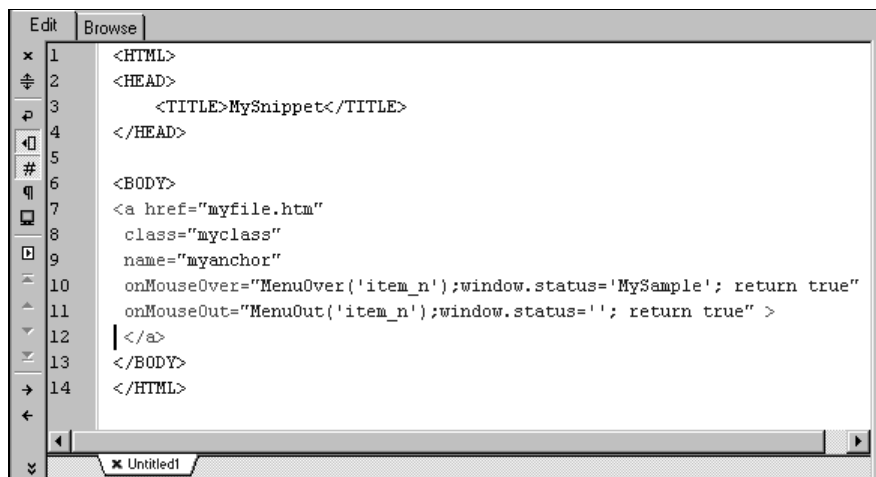


Рис. 4.30. Результат внедрения пользовательского фрагмента кода

Шаблоны кодов

В разд. "Настройки ColdFusion Studio" гл. 2 мы упоминали о шаблонах кодов.

Здесь давайте разберем создание собственного шаблона кода **Code Template** (Шаблон кода), с помощью которого можно решить ту же задачу, которая была рассмотрена в предыдущем разделе с применением **Snippets** (Фрагменты кода).

Итак, выбрав команду **Options | Settings** (Настройка | Параметры) или нажав клавишу <F8>, в появившемся диалоговом окне **Settings** (Параметры) активизируем раздел шаблонов кода **Editor | Code Templates** (Редактор | Шаблоны кода), где и следует воспользоваться кнопкой **Add** (Добавить) для добавле-

ния нового шаблона кода. Заполняем диалоговое окно для создания шаблона кода (рис. 4.31).

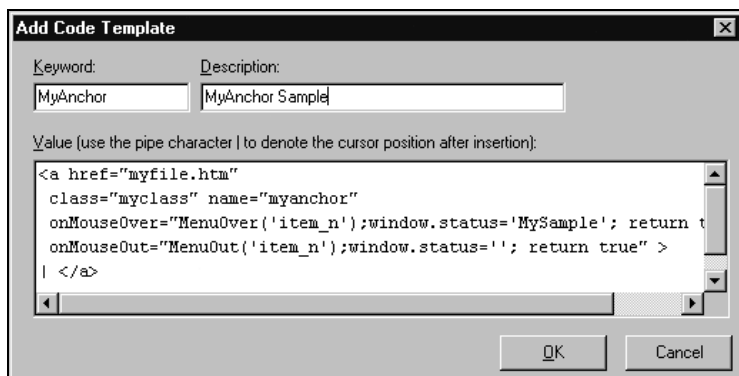


Рис. 4.31. Диалоговое окно добавления шаблона кода **Add Code Template**

Диалоговое окно для создания шаблона кода имеет помимо поля **Description** (Описание) также поле ввода **Keyword** (Ключевое слово). Что касается текстовой области для ввода блока кода **Value** (Значение), то ее следует рассматривать как один общий код. Однако этот код с легкостью можно разбить на две части, разделив текст кода символом вертикальной линии |. Данный символ указывает на местоположение текстового курсора.

Для использования шаблонов в режиме редактирования необходимо воспользоваться комбинацией клавиш быстрого доступа <Ctrl>+<J>. В этом случае откроется список существующих шаблонов кода, показанный на рис. 4.32.

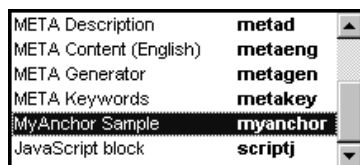


Рис. 4.32. Список шаблонов кода

И для внедрения кода останется лишь выбрать необходимый шаблон.

Базы данных

Окно для работы с базами данных располагается в панели **Resource** (Ресурс) в качестве вкладки **Database** (Базы данных) . Указанный модуль обеспечивает доступ к базам данных через поименованные источники данных.

Здесь вы можете просматривать данные, формировать запросы, сохранять запросы в качестве заготовок для повторного использования.

Если вы еще не установили соединение с RDS-сервером, то для работы с данным модулем это необходимо сделать. Соединение с RDS-сервером позволит получить список поименованных источников данных. Для этого достаточно из раскрывающегося списка, находящегося в верхней части окна данного модуля, выбрать единственный пункт **Add RDS Server** (Добавить RDS-сервер) и заполнить диалоговое окно конфигурации RDS-сервера **Configure RDS Server** (Конфигурация RDS-сервера).

В разд. "Установка соединения с удаленным сервером" гл. 2 подробно описан процесс установки RDS-сервера.

После конфигурирования RDS-сервера раскрывающийся список пополнится новым значением с названием добавленного сервера. На рис. 4.33 показано окно для работы с базами данных, содержащее список поименованных источников данных при успешном подключении к RDS-серверу **localhost**.

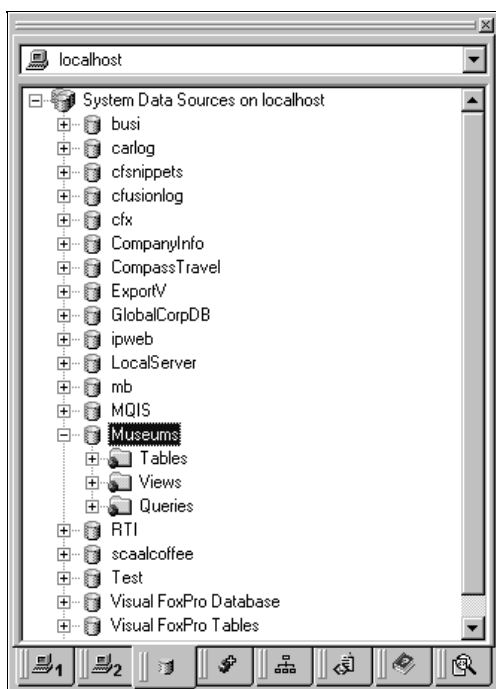


Рис. 4.33. Окно для работы с базами данных **Database**

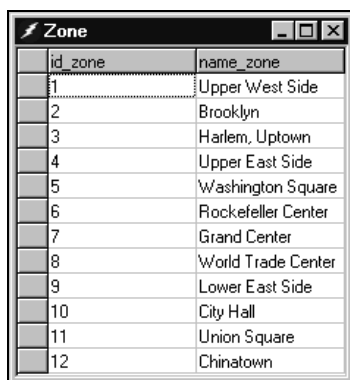
Обратите внимание, что каждый поименованный источник разделяется на три составляющие: **Tables** (Таблицы), **Views** (Представления) и **Queries** (Запросы). Что касается таблиц и представлений, то эти данные содержатся

непосредственно в базе данных, прописанной в поименованном источнике, а запросы ColdFusion Studio хранит в собственном каталоге в виде файлов с расширением vtm. Чуть позже продемонстрируем это.

Предварительно мы создали базу данных по музеям Нью-Йорка, что позволит наглядно продемонстрировать возможности данного ресурса. База данных музеев (Museums) состоит из трех связанных таблиц: одной основной Museum (Музей), и двух вспомогательных Species (Категория музея) и Zone (Зона размещения музея).

Просмотр данных

Для того чтобы осуществить просмотр данных как таблиц, так и представлений, достаточно выбрать таблицу или представление, щелкнуть правой кнопкой мыши и из появившегося контекстного меню выбрать команду **View Data** (Просмотр данных). На рис. 4.34 показано окно просмотра данных таблицы Zone.



id_zone	name_zone
1	Upper West Side
2	Brooklyn
3	Harlem, Uptown
4	Upper East Side
5	Washington Square
6	Rockefeller Center
7	Grand Center
8	World Trade Center
9	Lower East Side
10	City Hall
11	Union Square
12	Chinatown

Рис. 4.34. Окно просмотра данных таблицы Zone

В окне просмотра данных можно изменять ширину колонок и высоту строк.

Построение запроса

Запросы предназначены для получения выборок данных из одной или нескольких таблиц в наиболее удобной форме. Чаще всего такие выборки используются для анализа данных, построения по ним графиков и подготовки отчетов. Запросы также могут применяться в качестве добавления, обновления и удаления данных.

Для создания запроса необходимо выполнить следующие действия.

1. Выделите один из пунктов в поименованном источнике данных (это может быть как непосредственно источник данных, так и таблица, представ-

ление или запрос). Так или иначе, все запросы размещаются в категории **Queries** (Запросы) текущего источника данных.

2. Щелкните правой кнопкой мыши и из контекстного меню выберите команду **New Query** (Новый запрос).
3. Укажите в диалоговом окне **Select Table** (Выбор таблицы) таблицу или представление, на основании которого предполагается создать запрос. Причем, если выполнить команду создания запроса, предварительно выбрав одну из таблиц, то запрос будет создан автоматически на основе текущей таблицы.

На экране появится модальное окно для построения запроса **Query Builder** (Построитель запроса).

4. Определите характер запроса: **SELECT**, **INSERT**, **UPDATE** или **DELETE**, воспользовавшись панелью инструментов. По умолчанию указана выборка (**SELECT**) данных.
5. При необходимости сформируйте запрос из нескольких таблиц, воспользуйтесь одной из команд **Add Table** (Добавить таблицу) — по щелчку правой кнопкой мыши в пространстве визуального отображения таблиц и представлений — или **Add Tables** (Добавить таблицы) — используя панель инструментов или все ту же правую кнопку мыши. Такие добавления возможны только для запросов типа **SELECT** и **INSERT**.
6. Определите связь между таблицами, если в запросе используется более одной таблицы. Для этого достаточно зацепить указателем мыши поле одной из таблиц и, удерживая его, перетащить к полю другой таблицы. Результат тут же отобразится как в визуальной модели, так и в коде запроса, расположенного в нижней части окна построения запроса. Данное действие возможно только для запросов типа **SELECT** и **INSERT**.
7. При конкретном условии определите характер связи, щелкнув правой кнопкой мыши по графическому объекту, изображающему связь между таблицами, и выбрав необходимые условия (здесь подразумевается операция объединения **JOIN**). Данное действие возможно только для запросов типа **SELECT** и **INSERT**.
8. Выберите требуемые поля в столбце **Column** (Колонка) и установите необходимые параметры для каждого поля.
9. При желании предварительно просмотрите результат запроса на выборку, выполнив команду **Run Query** (Выполнить запрос), используя панель инструментов или правую кнопку мыши.
10. Сохраните запрос, выполнив команду **Save Query** (Сохранить запрос) с помощью панели инструментов или правой кнопки мыши.
11. Закройте модальное окно построителя запросов.

На рис. 4.35 приведен пример построения запроса с демонстрацией результата запроса.

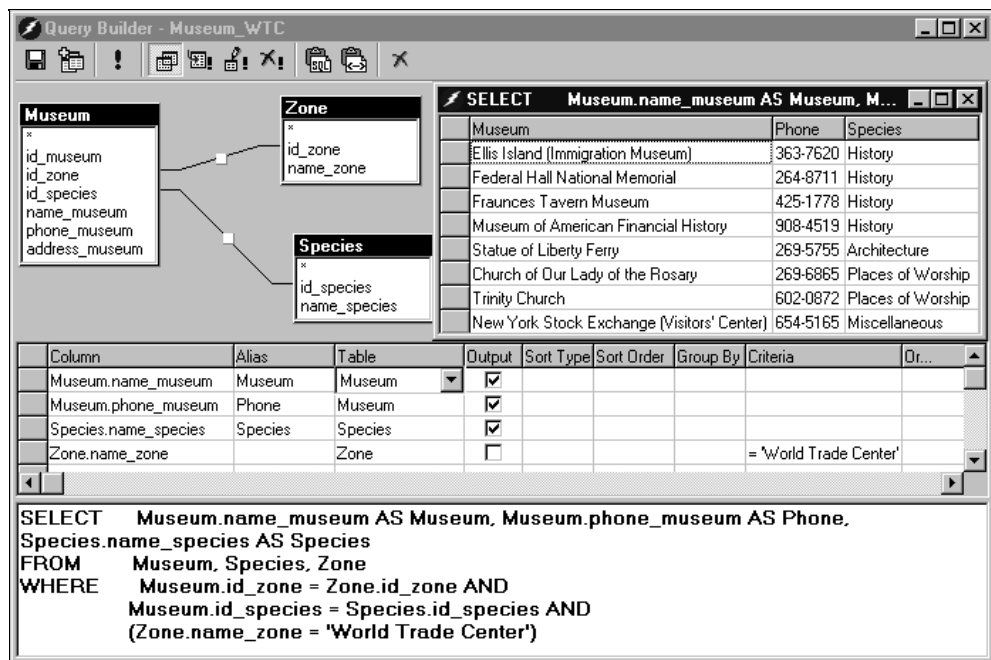


Рис. 4.35. Пример построения запроса с демонстрацией результата запроса

В данном примере мы производим выборку всех музеев, находящихся в районе World Trade Center, указывая название, телефон и категорию музея, а также для удобного отображения данных устанавливаем псевдоимена полей в столбце **Alias** (Псевдоним).

Использование запросов

Теперь, когда мы уже имеем представление о построении запросов, заострим свое внимание на том, каким образом можно использовать запросы.

Во-первых, созданный запрос всегда можно изменить, щелкнув по нему правой кнопкой мыши и выбрав команду **Edit Query** (Правка запроса). А во-вторых, после того как запрос уже пригоден для использования, его можно включать в код.

В модальном окне **Query Builder** (Построитель запроса) имеются два инструмента, предназначенные для копирования SQL-запросов.

Один инструмент — **Copy SQL to Clipboard** (Копировать SQL в буфер обмена)  — позволяет непосредственно копировать в буфер обмена SQL-запросы.

Другой инструмент — **Copy CFQUERY to Clipboard** (Копировать CF-запрос в буфер обмена)  — дает возможность копировать в буфер обмена SQL-запросы, помещенные внутрь тега `<CFQUERY>`, предназначенного для выполнения запросов посредством ColdFusion. (В главе 11 будет подробно описано использование тега `<CFQUERY>`.)

Впрочем, для включения SQL-запроса в код текущей страницы необязательно открывать окно построителя запроса, копировать запрос, а затем вставлять его из буфера обмена. Достаточно, удерживая указателем мыши значок этого запроса, перетащить его в рабочую область редактирования страницы. На рис. 4.36 показан результат внедрения SQL-запроса.

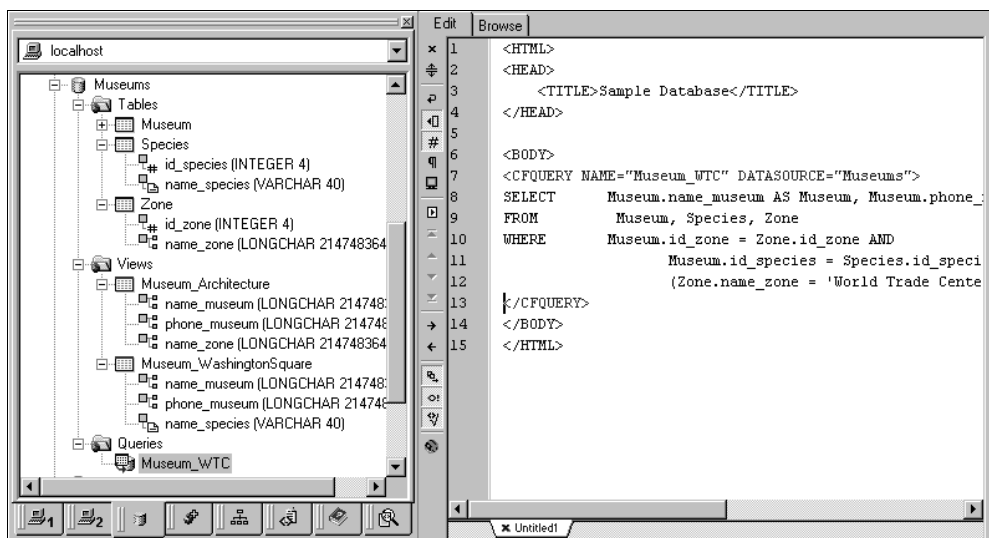


Рис. 4.36. Пример внедрения SQL-запроса в текущую страницу

В дальнейшем при желании внедренный код можно изменять вручную.

Уже было отмечено, что запросы хранятся в каталоге ColdFusion в виде файлов с расширением `vtm`. Воспользуемся стандартной службой поиска ОС Windows для того, чтобы отыскать созданный нами запрос под названием `Museum_WTC`. На рис. 4.37 продемонстрирован результат поиска.

На практике при разработке приложений приходится создавать достаточно сложные запросы и, следовательно, часто использовать команду `SELECT`, которая занимает центральное место в работе с данными.

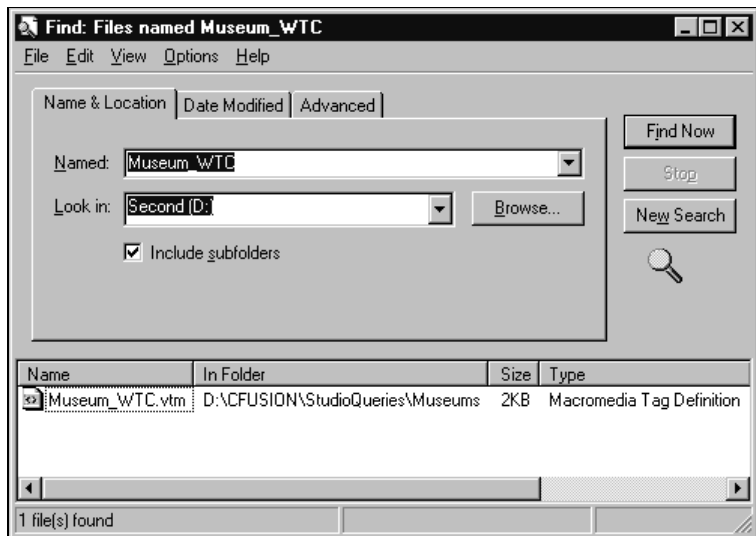


Рис. 4.37. Результат поиска созданного запроса

Отладка Web-страниц

В разд. "Панель отладки документов" гл. 3 мы уже подробно рассматривали каждый инструмент, включаемый в данную панель.

Теперь пришло время продемонстрировать работу с инструментами для отладки Web-страниц на простом примере.

Итак, предположим мы уже создали страницу `welcome.cfm` и, не усложняя ситуацию, установим только одну точку прерывания.

1. Для этого необходимо поместить текстовый курсор в ту строку кода, где предполагается установить эту точку и, воспользовавшись переключателем состояния **Toggle Breakpoint** (Точка прерывания) , установить точку прерывания.

Точка прерывания будет отмечена в левом вертикальном поле специальным символом, а строка выделена красным цветом (рис. 4.38).

2. Далее следует начать отладку текущей Web-страницы, используя инструмент **Start/Continue** (Начать/Продолжить)  панели отладки документов.
3. Затем откроется диалоговое окно **Remote Development Settings** (Параметры удаленной разработки) на вкладке стартовых настроек отладки документа **Debug Start** (Запуск отладки), позволяющей изменять RDS-сервер отладки и путь к начальной странице (рис. 4.39).

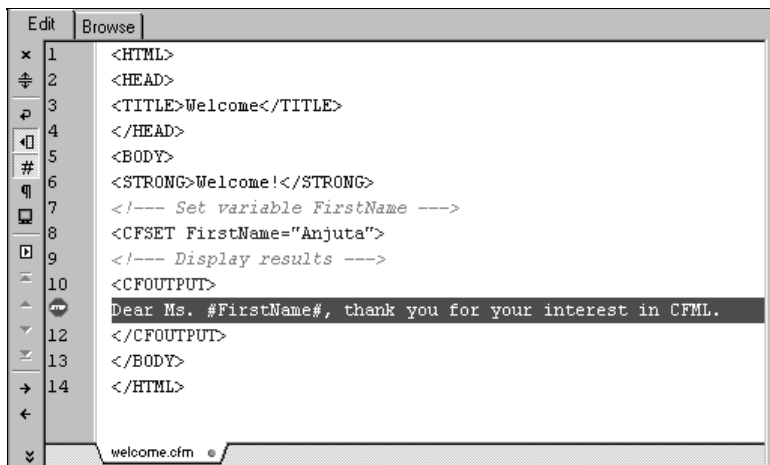
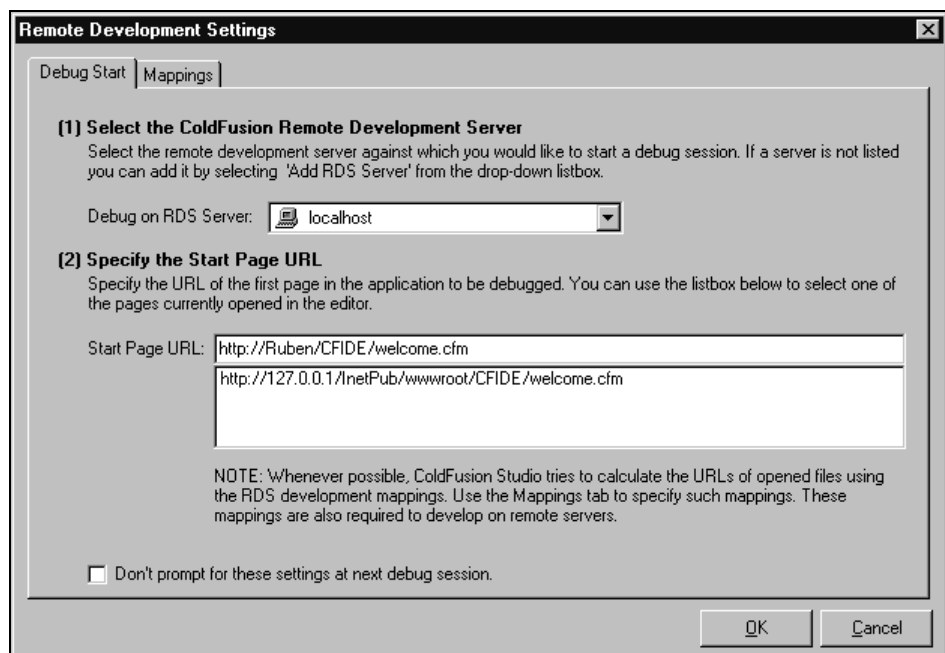


Рис. 4.38. Установка точки прерывания

Рис. 4.39. Диалоговое окно **Remote Development Settings**, вкладка **Debug Start**

4. После подтверждения начала выполнения отладки страницы нажатием кнопки **ОК** откроется окно отладки **Debug** (Отладка) с набором вкладок, а на панели отладки документов активизируются инструменты перехода по точкам прерывания и инструмент окончания сессии отладки.

Рассмотрим каждую вкладку окна отладки документа и начнем с переменных — **Variables** (Переменные). Данная вкладка позволяет просматривать список локальных переменных (рис. 4.40).

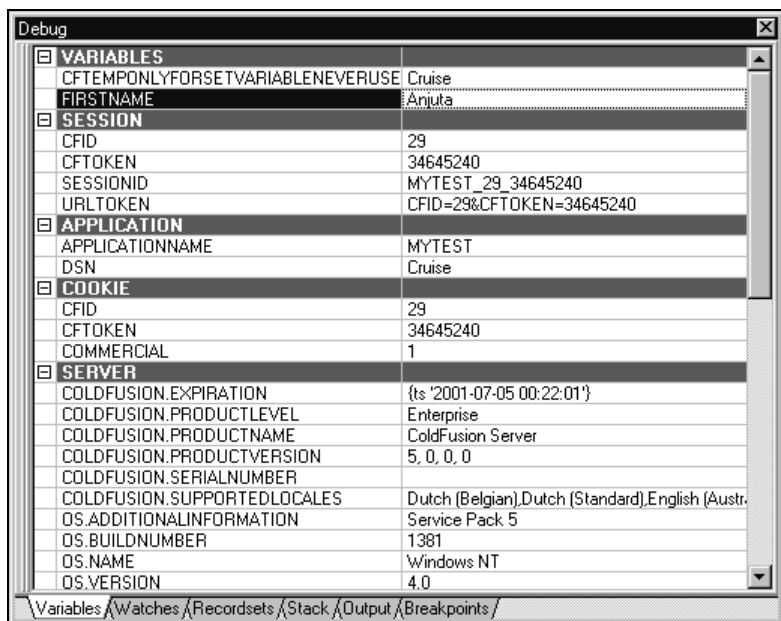


Рис. 4.40. Окно **Debug**, вкладка **Variables**

Следующая категория **Watches** (Просмотр) позволяет просматривать значения, которые принимают выражения во время работы программы. В нашем примере до точки прерывания была объявлена переменная `FirstName`. В связи с этим мы смело можем набрать в верхнем поле раскрывающегося списка имя этой переменной и выполнить команду просмотра значения, используя кнопку **Watch** (Просмотр) — рис. 4.41.

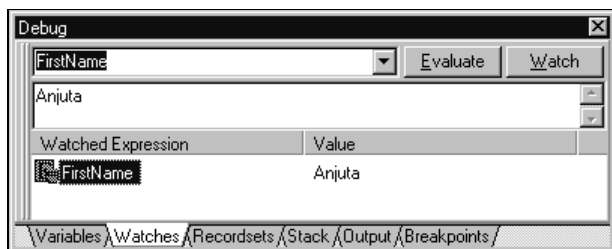


Рис. 4.41. Окно **Debug**, вкладка **Watch**

Выражение, помещенное в список, можно удалять, предварительно выделив его и нажав на клавишу <Delete>.

Так как в нашем примере не используется доступ к базам данных, следовательно, и вкладка просмотра списка набора данных **Recordsets** (Наборы данных) является пустой.

Вкладка **Stack** (Стек) позволяет просматривать выполняемый тег (рис. 4.42).



Рис. 4.42. Окно **Debug**, вкладка **Stack**

Вкладка **Output** (Вывод) дает возможность просматривать результат выполнения активной страницы (рис. 4.43).

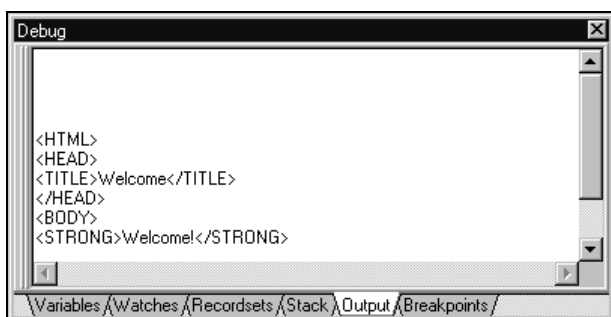


Рис. 4.43. Окно **Debug**, вкладка **Output**

И вкладка **Breakpoints** (Точки прерывания) уже, судя по названию, позволяет просматривать список точек прерывания. В нашем случае такая точка одна (рис. 4.44).



Рис. 4.44. Окно **Debug**, вкладка **Breakpoints**

Далее, если бы у нас было установлено несколько подобных точек прерывания, можно было бы, воспользовавшись активными инструментами перехода по точкам, шаг за шагом, точка за точкой отлаживать текущую страницу, тщательно просматривая отображаемые данные в окне отладки **Debug** (Отладка).

Завершить этап отладки текущей Web-страницы можно, воспользовавшись инструментом окончания сессии отладки **End Debug Session** (Завершить сеанс отладки)  или нажав комбинацию клавиш <ALT>+<F5>.

Редактор стилей

Приложение ColdFusion Studio включает в себя редактор каскадных таблиц стилей (Cascading Style Sheets, CSS) — TopStyle. Этот редактор невероятно удобен в использовании и поддерживает CSS-спецификации (CSS 1.0 и CSS 2.0), а также поддерживает различные расширения, встроенные в Internet Explorer версий 3, 4, 5, Netscape Communicator 4 и в Opera 3.5.

Запустить редактор TopStyle можно, нажав кнопку **Style Editor** (Редактор стиля) , расположенную на панели вспомогательных средств **Tools** (Сервис).

Для редактирования уже существующего набора стилей необходимо вызвать контекстно-зависимое меню, щелкнув правой кнопкой мыши по фрагменту кода в рамках пары тегов <STYLE></STYLE>, и выбрать команду **Edit Style Block** (Правка блока стиля).

Каскадные таблицы стилей могут быть определены как в отдельных файлах с расширением css для последующего подключения к Web-страницам, так и в пределах кода страницы. Они позволяют форматировать текст, таблицы и иные элементы Web-страниц. Тематика книги не позволяет подробно останавливаться на описании принципов формирования каскадных таблиц стилей, приведем лишь пример форматирования одного из стилей `txtprompt` с использованием редактора. На рис. 4.45 показано окно редактора TopStyle.

Как видно из рис. 4.45, окно редактора TopStyle разбито на три составляющие: рабочая область для написания кода, область предварительного просмотра (**Preview**) и инспектор стилей (**Style Inspector**). Для создания собственного стиля достаточно в рабочей области редактора записать наименование стиля (начиная с точки, как показано на рисунке) и открыть фигурную скобку. Закрывающая фигурная скобка вставляется автоматически, после чего воспользовавшись инспектором стилей, остается установить необходимые свойства. Например, цвет (`color`), название шрифта (`font-family`), размер шрифта (`font-size`) и т. д. Область предварительного просмотра позволяет наглядно просматривать полученный результат.

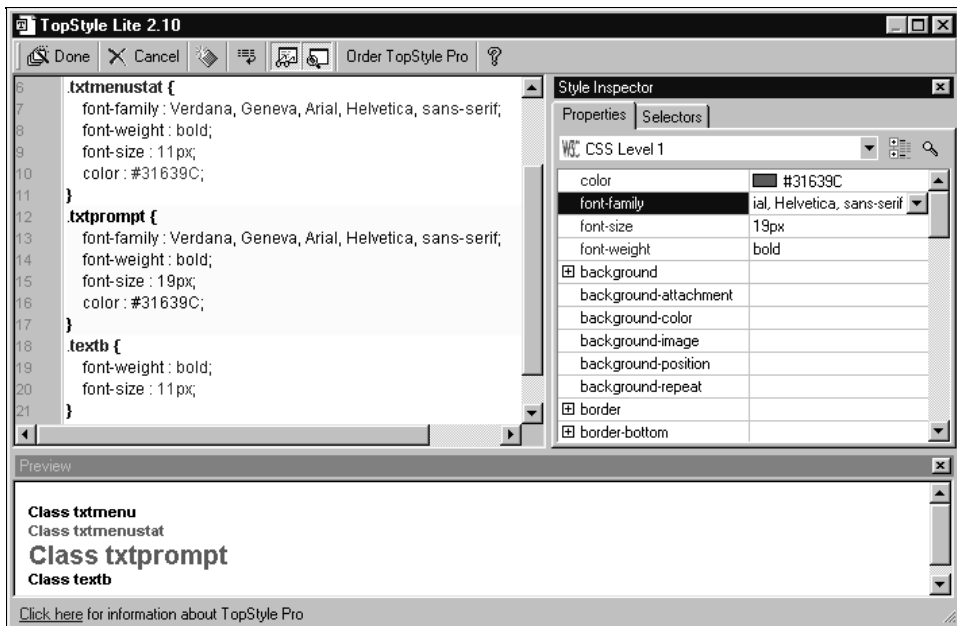


Рис. 4.45. Окно редактора TopStyle

Причем, как бы вы не изменяли код каскадных таблиц стилей с использованием TopStyle, всегда есть возможность в верхнем меню редактора выбрать команду **Cancel** (Отмена), что позволит вернуться в область ColdFusion Studio без изменений. Команда **Done** (Завершить) все из того же меню редактора позволяет применить совершенные вами изменения.

Для назначения сформированного нами стиля какому-либо фрагменту текста страницы необходимо определить так называемый класс (class) для этого фрагмента. Например, определим для тега абзаца класс "txtprompt".

```
<P class="txtprompt" align="right">
```

```
    New York's BEST Sightseeing and Live Music Cruises
```

```
</P>
```

Очевидно, что в этом случае текст, заключенный между парой тегов `<P></P>`, будет отформатирован в соответствии с указанным классом "txtprompt".

Редактор Image Map

Приложение ColdFusion Studio содержит достаточно удобный редактор для создания графических ссылок с разбивкой рисунков на области (area). Сформированные области карты можно увеличивать, уменьшать, перетаски-

вать и удалять. При использовании редактора Image Map код получается не- избыточным и удобным.

Для создания новой карты:

1. Необходимо нажать кнопку **New Image Map** (Новое изображение-карта) , расположенную на панели вспомогательных средств **Tools** (Сервис).
2. В появившемся диалоговом окне **Create Image Map** (Создать изображение-карту) следует выбрать рисунок и указать название карты **Map Name** (Имя карты). Список рисунков в данном диалоговом окне формируется из уже помещенных графических ссылок в код текущей страницы. Однако в этом же диалоговом окне можно выбрать другой рисунок из GIF- и JPEG-файлов, расположенных на вашем компьютере, вследствие чего помимо самой карты областей, будет сформирован тег графического изображения, основанного на указанном рисунке.

К примеру, подготовив заранее рисунок и задав название карты MyMap, мы можем открыть окно **Image Map Editor** (Редактор Image Map), показанное на рис. 4.46.

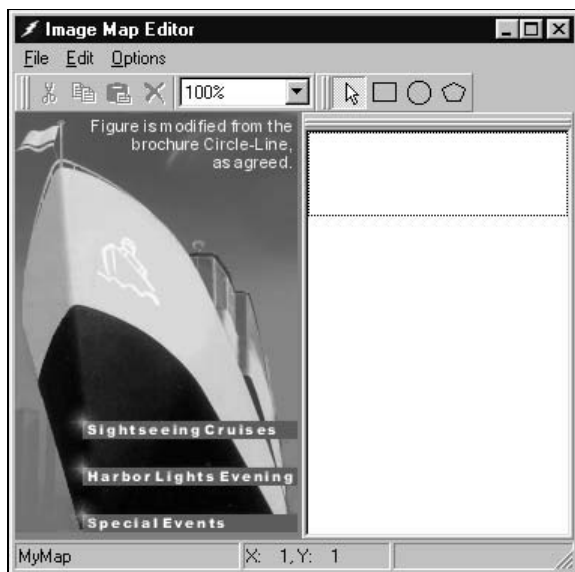


Рис. 4.46. Окно редактора Image Map Editor

Окно редактора Image Map разбито на две части — непосредственно сам рисунок и список областей.

3. Следующим шагом нужно выбрать области ссылок, для чего, собственно говоря, все это и задумано. Для более точного выбора области можно прибегнуть к инструменту масштабирования, расположенному в верхней

части редактора. В нашем примере необходимые области для выделения имеют прямоугольную форму, поэтому для выделения выбираем инструмент прямоугольной формы.

- После выделения необходимой области открывается диалоговое окно редактора тега `<AREA>`, где нужно выбрать страницу, на которую и следует создать ссылку. На рис. 4.47 показано окно редактора тега `<AREA>`.

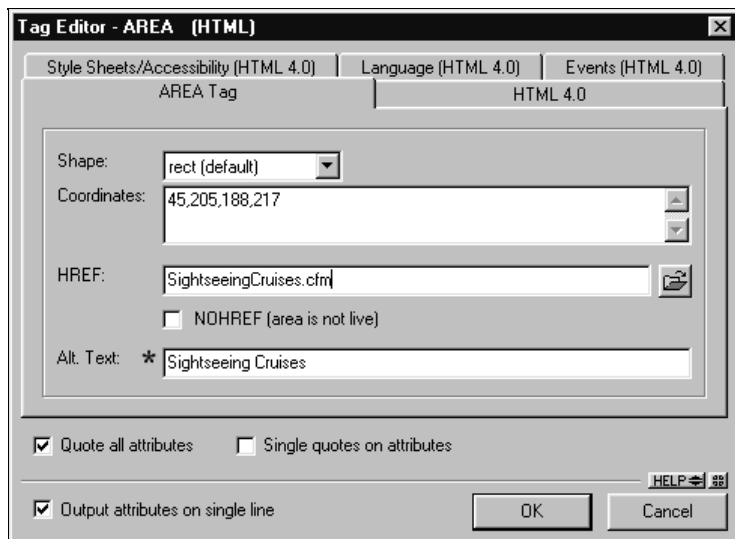


Рис. 4.47. Окно редактора тега `<AREA>`

Таким образом, можно создавать целый ряд областей различной формы. После повторения подобных манипуляций мы создали еще две области для нашего примера. Результат продемонстрирован на рис. 4.48.

- Остается лишь после выхода из редактора либо сохранить изменения, либо отвергнуть их — это уже на ваше усмотрение.

При сохранении созданных нами областей код выглядит следующим образом:

```
<MAP name="MyMap">
  <AREA alt="Sightseeing Cruises" coords="45,205,188,217"
    href="SightseeingCruises.cfm">
  <AREA alt="Harbor Lights Evening" coords="45,236,188,248"
    href="HarborLightsEvening.cfm">
  <AREA alt="Special Events" coords="45,267,188,280"
    href="SpecialEvents.cfm">
</MAP>
```

При этом к существующему тегу графического изображения был добавлен атрибут `usemap` со значением `"#MyMap"`:

```
<IMG src="pic1.gif" alt="" border="0" usemap="#MyMap">
```



Рис. 4.48. Пример работы в редакторе Image Map Editor

Приятно, что для редактирования уже существующей карты областей ссылок нужно всего лишь вызвать контекстно-зависимое меню, щелкнув правой кнопкой мыши по фрагменту кода в рамках пары тегов `<MAP></MAP>`, и выбрать команду **Edit Image Map** (Правка изображения-карты).

Все великолепие редактора для создания графических ссылок с разбивкой изображения на части может понять тот, кому хотя бы однажды довелось писать подобные коды вручную, особенно при существовании многоугольников неправильной формы, например, в контурных картах.

Работа с буфером обмена

Как и любое приложение, ColdFusion Studio позволяет сохранять текст в буфере обмена. Но при этом данная функция обеспечивает копирование множества текстовых блоков и сохранение их в качестве отдельных записей в буфере обмена. На рис. 4.49 изображена панель редактирования **Edit** (Правка), где расположены инструменты, позволяющие вставлять фрагменты из буфера обмена, показывать его содержимое для выборочной вставки, вставлять все фрагменты из буфера обмена и очищать его.

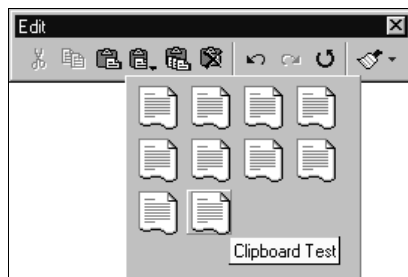


Рис. 4.49. Пример работы с набором фрагментов буфера обмена

В буфер можно помещать до 36 фрагментов, на взгляд автора, это более чем достаточно. Но при необходимости данную настройку можно изменить, выполнив команду **Option | Settings** (Настройка | Параметры), перейдя к категории **Editor** (Редактор) и изменив значение **Maximum clipboard entries** (Максимальное количество записей в буфере обмена). Когда количество записей в буфере начинает превышать установленное в настройках значение, последнее копирование удаляет старейшую запись и добавляет новую в конец буферного пространства.

Восстановление резервных копий

В ColdFusion Studio встроена служба автоматического резервирования документов. Если у вас возникает необходимость восстанавливать содержимое утерянных по какой-либо причине документов, то службу автоматического резервирования (**Auto-Backup**) следует держать включенной. Включение и отключение автоматического резервирования осуществляется за счет флажка **Enable Auto-Backup** (Включить резервное копирование), расположенного в разделе **Editor | Auto-Backup** (Редактор | Резервное копирование), получить доступ к которому можно, выбрав команду **Options | Settings** (Настройка | Параметры) или нажав клавишу <F8>. В этом же разделе вы можете регулировать расположение резервируемых документов **Backup Directory** (Каталог резервного копирования), способ сохранения, количество дней хранения резервируемых документов перед удалением и промежуток времени резервирования документов (**Timed Auto-Backup**), по умолчанию равный 10 минутам.

Для восстановления резервных копий утерянных файлов следует выбрать команду **Options | Auto-Backup File Maintenance** (Настройка | Восстановить резервную копию). Затем, отметив необходимые документы из списка, содержащего автоматически зарезервированные файлы в процессе определенного времени работы (по умолчанию 10 дней), следует выполнить команду **File | Open Selected Files for Edit** (Файл | Открыть выбранные файлы для редактирования) или нажать комбинацию клавиш <Ctrl>+<O>. Можно также,

предварительно отметив необходимые документы, воспользоваться панелью инструментов, нажав кнопку, изображающую папку с документами, как показано на рис. 4.50.

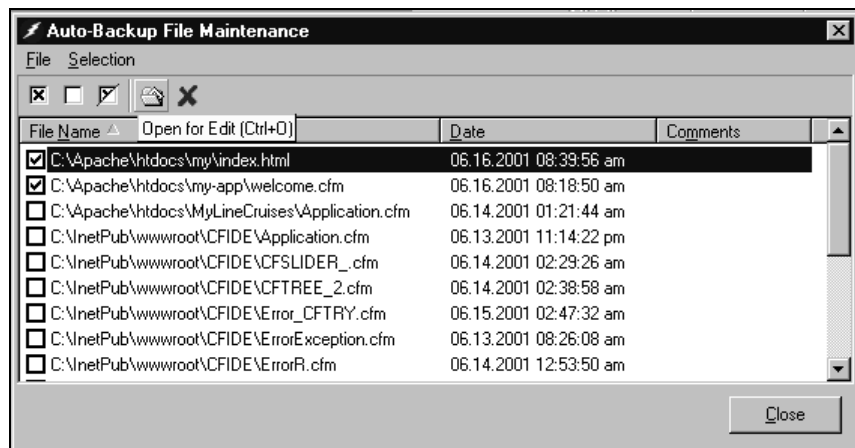


Рис. 4.50. Окно со списком зарезервированных файлов
Auto-Backup File Maintenance

В результате выполненных действий выбранные документы будут открыты для редактирования, причем, в конце названия каждого документа добавляется надпись "(backup)". К примеру, при восстановлении документа welcome.cfm в редакторе ColdFusion Studio будет открыт документ с названием "welcome.cfm (backup)", который впоследствии имеет смысл сохранить под первоначальным названием, т. е. welcome.cfm. На самом же деле все зарезервированные файлы имеют имя в формате:

имя_файла + три_цифры_по_возрастанию + расширение_файла

Для нашего примера документ welcome.cfm может храниться в файловой системе как welcome000.cfm. Не исключено, что там же может находиться файл welcome001.cfm. Это говорит лишь о том, что вы используете документы с одинаковыми именами в разных проектах. И файл welcome000.cfm, и файл welcome001.cfm будут иметь различные источники, в чем можно убедиться визуально, используя как вышеупомянутое окно **Auto-Backup File Maintenance** (Восстановление резервной копии), так и просмотрев файл AutoBackup.ini, применяемый в качестве журнала резервных файлов и находящийся по умолчанию в каталоге \ColdFusion Studio 5\AutoBackup.

Для удобства работы с флажками, находящимися слева от каждого резервного файла (см. рис. 4.50), существуют следующие команды:

☐ **Selection | Select All** (Выделение | Выделить все) — предполагает выделение сразу всех файлов;

- ☐ **Selection | Unselect All** (Выделение | Снять выделение всех) — позволяет убрать выделение для всех файлов;
- ☐ **Selection | Toggle Checks for Selected Files** (Выделение | Только выделенные файлы) — переключатель, позволяющий включать или отключать только выделенные файлы.

Впрочем, если вы уверены, что та или иная резервная копия файла вряд ли когда-либо понадобится, ее можно намеренно удалить с помощью команды **File | Remove Selected Files** (Файл | Удалить выделенные файлы) или нажать клавишу <Delete>. Не забудьте предварительно отметить переключатели рядом с теми файлами, которые впоследствии будут удалены. Команда удаления так же, как и команда восстановления, дублируется на панели инструментов и реализуется в виде кнопки с изображением перечеркнутых линий красного цвета.

На этом мы заканчиваем краткое описание работы в среде ColdFusion Studio и плавно переходим к следующей главе, посвященной проектированию Web-приложений.

Глава 5



Проектирование Web-приложения

Вне зависимости от проекта процесс разработки программного обеспечения проходит одни и те же основные стадии: исследование требований к системе, постановка задачи, выбор инструментальных средств, кодирование, тестирование работающей системы и ее сопровождение. Что касается инструментальных средств, то наш выбор остановлен на ColdFusion.

Постановка задачи

Предположим, нам поступил заказ на разработку сайта — круизы и экскурсии по достопримечательностям любимого города. Постановка задачи требует ответов на целый ряд вопросов.

- ☐ Каково назначение вашего приложения?

Ответ: электронный бизнес в культурно-развлекательной сфере.

Знание общей задачи приложения помогает не отклоняться от заданной цели. Ответ на этот вопрос поможет установить критерии для оценки требований к функциональным возможностям продукта.

- ☐ Какова непосредственная цель?

Ответ: привлечение туристов, ознакомление с возможными круизами и экскурсиями, бронирование и распространение билетов.

- ☐ Каков состав аудитории?

Ответ: представители общественности, имеющие желание и возможность путешествовать.

Существуют разные способы построения базы данных для "мировой паутины". Знание состава аудитории позволит решить, какие меры безопасности принять, какие возможности клиентской части использовать и т. п.

- ☐ Кто ваши конкуренты?

Здесь мы не дадим вам четкий ответ, т. к. это могут быть разные порталы или специализированные сайты, предлагающие подобные услуги. От ответа на этот вопрос многое зависит.

❑ Какая информация публикуется?

Ответ: таблицы расписаний, примечание по каждому круизу и экскурсии с фотографиями и пояснениями, цены, реклама туристических агентств, прогноз погоды, формы заказов и т. п.

Существует еще целый ряд дополнительных вопросов, ответив на которые можно получить более полную картину будущего приложения. Например, какая потребуется пропускная способность? Производится ли наращивание базы данных при помощи Web-интерфейса? Кто будет отвечать за информационное содержимое? Кто будет отвечать за поддержание работоспособности Web-сервера? Каковы требования к безопасности? На что следует обратить внимание при тестировании? И т. д.

Бизнес-правила

Прежде чем разработать базу данных, следует сформулировать правила, позволяющие уточнить общие требования к информационной модели. Назовем такие правила *бизнес-правилами*, которые в нашем случае могут быть следующими:

1. Стоимость круизов или экскурсий не зависит от времени суток, дней недели и времени года.
2. Стоимость круизов и экскурсий зависит от возраста туристов (взрослые, дети и престарелые люди).
3. Один и тот же маршрут обслуживается однотипным транспортным средством.
4. Посадочные места не нумеруются и резервируются по мере появления пассажиров.
5. Периодичность совершения круизов и экскурсий зависит от дней недели (будни, выходные и праздники) и времени года (сезонный режим).
6. Все круизы и экскурсии, совершаемые в рабочие дни недели, дублируются в выходные и праздничные дни, но круизы и экскурсии, запланированные в выходные и праздничные дни, необязательно совершаются в рабочие дни.
7. За два часа до отправления бронирование билетов прекращается.
8. Возвращение в продажу забронированных билетов заканчивается за час до отправления.
9. При регистрации электронный адрес является входным именем пользователя (Login ID).
10. При бронировании билетов все денежные расчеты производятся через кредитные карточки в соответствии с правилами банковских операций. Для этой цели используются услуги сторонней фирмы.

Теперь, когда задача определена, сформулированы бизнес-правила, можно начать разработку базы данных.

Разработка базы данных

Процесс создания информационной модели начинается с формулирования концептуальных требований, которые могут определяться также для некоторых задач, в ближайшее время не планируемых для реализации. Это может несколько повысить трудоемкость работы, однако поможет наиболее полно учесть все нюансы функциональности, требуемой для разрабатываемой системы, и снизить вероятность ее переделки в дальнейшем.

Концептуальная модель представляет объекты и их взаимосвязи без указания способов их физического хранения.

Таким образом, концептуальная модель является, по существу, моделью предметной области. При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели основано на анализе решаемых задач обработки данных. Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных. Здесь имеются в виду данные, используемые как в уже разработанных прикладных программах, так и в тех, которые только будут реализованы.

Концептуальная модель транслируется затем в *модель данных*, совместимую с выбранной СУБД. Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБД. Это потребует изменения концептуальной модели. Версия концептуальной модели, которая может быть обеспечена конкретной СУБД, называется *логической моделью*.

Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения. Логическая модель данных может быть реляционной, иерархической или сетевой. Пользователем выделяются подмножества этой логической модели, называемые *внешними моделями* (в некоторых источниках их также называют *подсхемами*), отражающие их представления о предметной области. Внешняя модель соответствует представлениям, которые пользователи получают на основе логической модели. Логическая модель отображается в физическую память, такую, как диск, лента или иной носитель информации.

Физическая модель, определяющая размещение данных, методы доступа и технику индексирования, называется внутренней моделью системы.

Внешние модели никак не связаны с типом физической памяти, в которой будут храниться данные с методами доступа к этим данным. Это положение отражает *первый уровень* независимости данных. С другой стороны, если концептуальная модель способна учитывать расширение требований к системе в будущем, то вносимые в нее изменения не должны оказывать влияния на существующие внешние модели. Это *второй уровень* независимости данных. Важно помнить, что построение логической модели обусловлено требованиями используемой СУБД.

Все актуальные требования предметной области и адекватные им "скрытые" требования на стадии проектирования должны найти свое отражение в концептуальной модели. Конечно, нельзя предусмотреть все возможные варианты использования и изменения базы данных. Но в большинстве предметных областей такие основные данные, как объекты и их взаимосвязи, относительно стабильны. Меняются только информационные требования, т. е. способы использования данных для получения информации.

Степень независимости данных определяется тщательностью проектирования базы данных. Всесторонний анализ объектов предметной области и их взаимосвязей минимизирует влияние изменения требований к данным в одной программе на другие программы. В этом и состоит всеобъемлющая независимость данных.

Основное различие между указанными выше тремя типами моделей данных (концептуальной, логической и физической) состоит в способах представления взаимосвязей между объектами. При проектировании БД мы воспользуемся одним из CASE-инструментов.

Итак, будем последовательны, в результате анализа поставленной задачи и обработки требований составим концептуальную модель. Начнем с определения объектов (сущностей) и взаимосвязей между ними.

Перечень сущностей:

- ☐ круизы (экскурсии) — TOUR;
- ☐ промежуточные пункты назначения — DESTINATION;
- ☐ маршруты — ROUTE;
- ☐ расписание — SCHEDULE;
- ☐ сезоны — SEASON;
- ☐ бронированные билеты — TICKET;
- ☐ клиенты (путешественники) — TRAVELER.

На рис. 5.1 представлена упрощенная схема информационной модели с определением взаимосвязей.

На следующем этапе зададим атрибуты каждой сущности и укажем первичные ключи. Не будем углубляться в теорию построения базы данных, ска-

жем только, что этого требуют правила построения базы данных. На рис. 5.2 представлена концептуальная модель базы данных CRUISE.

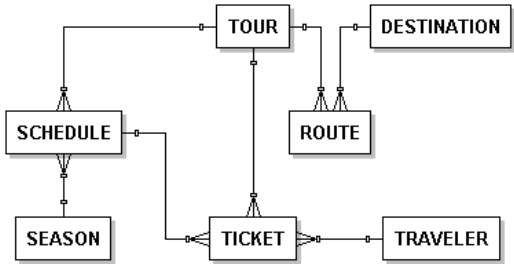


Рис. 5.1. Упрощенная схема информационной модели с определением взаимосвязей

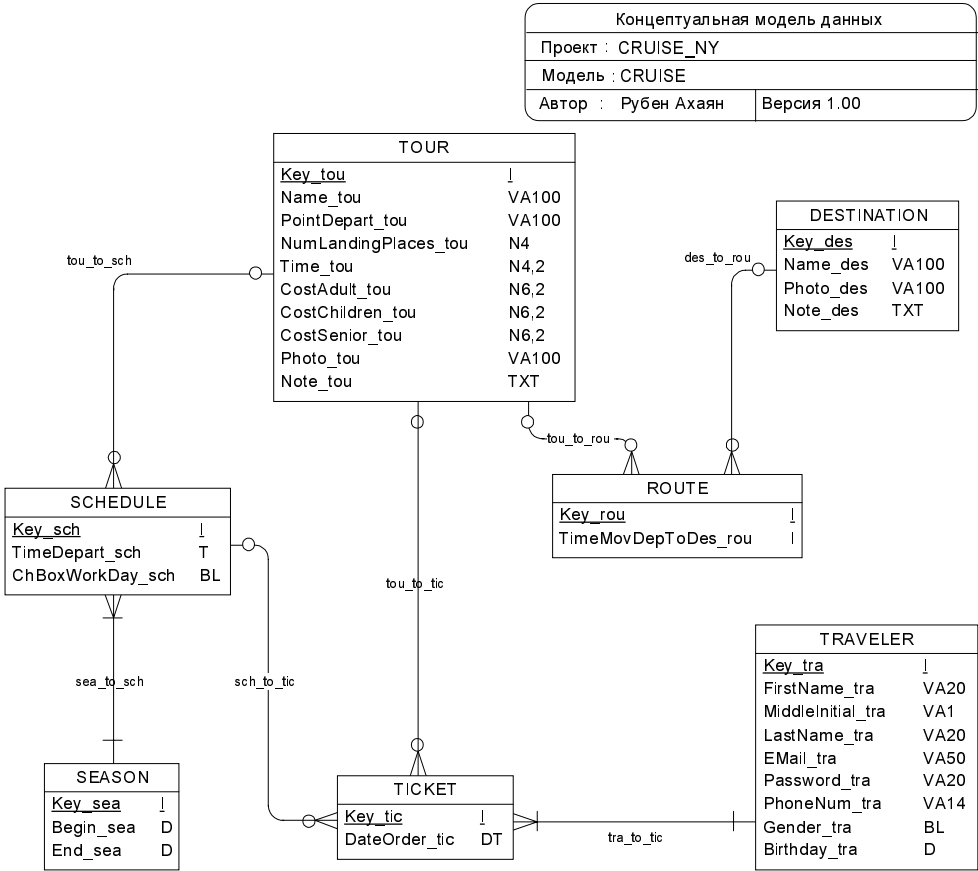


Рис. 5.2. Концептуальная модель базы данных CRUISE

Наступило время привести модель к требуемому уровню нормальной формы. Приведение модели к требуемому уровню нормальной формы является основой построения реляционной базы данных и представляет собой набор критериев, которыми следует руководствоваться при проектировании реляционной БД. Перечислим три нормальные формы таблиц.

Первая нормальная форма означает такую структуру таблицы, в которой не могут находиться повторяющиеся поля данных.

Вторая нормальная форма имеет отношение к строкам, в которых каждый столбец функционально зависит от первичного ключа.

Третья нормальная форма имеет отношение к тем записям, которые соответствуют второй нормальной форме и, кроме того, не содержат каких-либо транзитивных зависимостей. Транзитивная зависимость возникает, когда в таблице находится столбец, который, не являясь ключом, описывает другие столбцы.

Сформулируем основные правила, которым нужно следовать при построении базы данных.

- ❑ *Исключайте повторяющиеся группы.* Для каждого набора связанных атрибутов создайте отдельную таблицу и снабдите ее первичным ключом. Выполнение этого правила автоматически приведет ко второй нормальной форме.
- ❑ *Исключайте избыточные данные.* Если атрибут зависит только от части составного ключа, переместите атрибут в отдельную таблицу. Это правило помогает избежать потери одних данных при удалении каких-то других. Везде, где возможно использование идентификаторов вместо описания, выносите в отдельную таблицу список идентификаторов с пояснениями к ним.
- ❑ *Исключайте столбцы, которые не зависят от ключа.* Если атрибуты не влияют на описание ключа, переместите их в отдельную таблицу.

С учетом вышеизложенного переведем нашу концептуальную модель в физическую. Причем используемый нами CASE-инструмент PowerDesigner DataArchitect корпорации Sybase позволяет генерировать физическую модель автоматически для большинства серверов баз данных. Отдавая дань корпорации Sybase, сгенерируем нашу базу под Sybase SQL Anywhere. На рис. 5.3 представлена физическая модель базы данных CRUISE.

Чаще всего на этапе физического проектирования производятся меры по обеспечению целостности базы данных. В СУБД целостность данных обеспечивается набором специальных предложений, называемых ограничениями целостности. Ограничение целостности — это набор определенных правил, которые устанавливают допустимость данных и связей между ними.

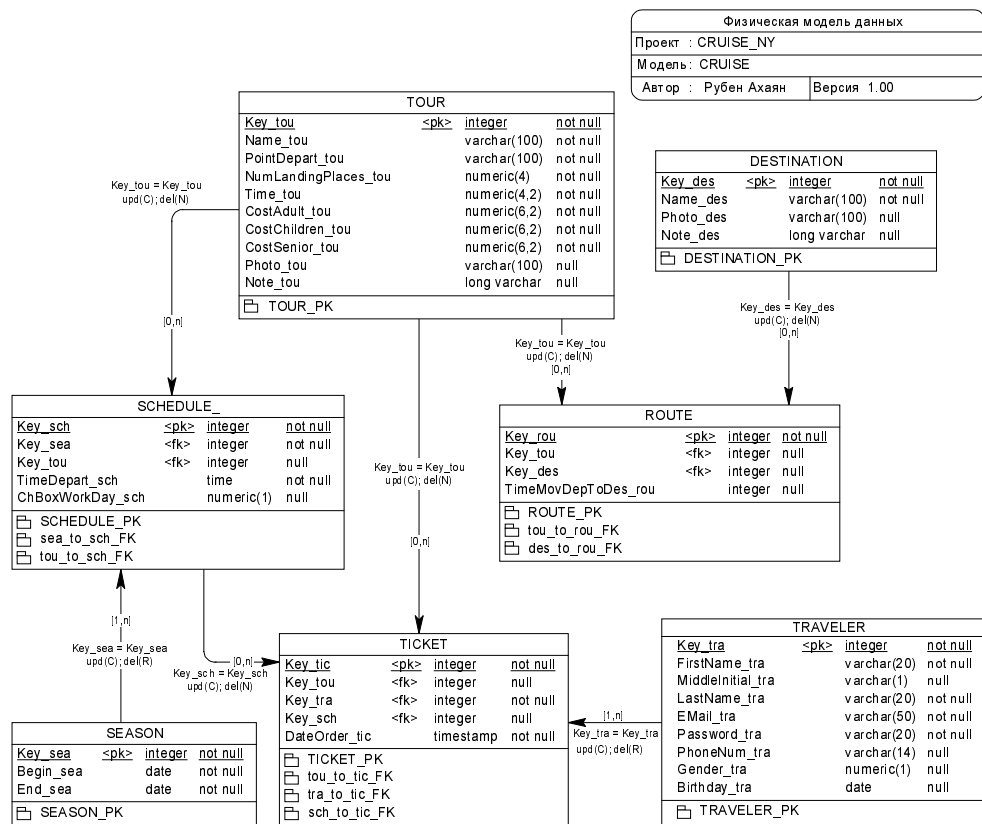


Рис. 5.3. Физическая модель базы данных CRUISE

Ограничения целостности, в свою очередь, могут относиться к разным объектам БД: атрибутам (полям), записям, отношениям, связям между ними и т. д. Для реализации ограничений целостности, имеющих отношение к записи, таблицам или связям между ними, в СУБД используются триггеры. Как конкретно реализуются подобные триггеры, мы не будем рассматривать отдельно и оставим на самостоятельное изучение.

Физическое описание модели

В следующих разделах приведен отчет для созданной нами базы данных CRUISE.

Приведем сокращения:

- P — Primary key (первичный ключ);
- M — Mandatory (принудительная связь);
- F — Foreign key (внешний ключ);

- ☐ C — Cluster (кластеризация);
- ☐ U — Unique (уникальность).

Таблицы (Tables)

В табл. 5.1—5.3 приведено описание таблицы DESTINATION.

Таблица 5.1. Список полей для таблицы DESTINATION

Имя поля	Описание	Тип данных	P	M
Key_des	Key destination (Уникальный ключ пункта назначения)	Integer	Да	Да
Name_des	Name destination (Наименование пункта назначения)	Varchar(100)	Нет	Да
Photo_des	Photo destination (Фотография пункта назначения)	Varchar(100)	Нет	Нет
Note_des	Note destination (Примечание)	Long varchar	Нет	Нет

Таблица 5.2. Список индексов для таблицы DESTINATION

Название индекса	P	F	U	C	Индексируемое поле	Сортировка
DESTINATION_PK	Да	Нет	Да	Нет	Key_des	ASC (По возрастанию)

Таблица 5.3. Список связей для таблицы DESTINATION

Связь для таблицы	Первичный ключ	Вторичный ключ
ROUTE	Key_des	Key_des

В табл. 5.4—5.6 приведено описание таблицы ROUTE.

Таблица 5.4. Список полей для таблицы ROUTE

Имя поля	Описание	Тип данных	P	M
Key_rou	Key route (Уникальный ключ маршрута)	Integer	Да	Да
Key_tou	Key tour (Ключ круиза)	Integer	Нет	Нет
Key_des	Key destination (Ключ пункта назначения)	Integer	Нет	Нет

Таблица 5.4 (окончание)

Имя поля	Описание	Тип данных	Р	М
TimeMovDep ToDes_rou	Time of moving from a point of departure up to destination (minute) (Время перемещения от пункта отправления до текущего пункта в минутах)	Integer	Нет	Да

Таблица 5.5. Список индексов для таблицы ROUTE

Название индекса	Р	F	U	C	Индексируемое поле	Сортировка
ROUTE_PK	Да	Нет	Да	Нет	Key_rou	ASC (По возрастанию)
tou_to_rou_FK	Нет	Да	Нет	Нет	Key_tou	ASC (По возрастанию)
des_to_rou_FK	Нет	Да	Нет	Нет	Key_des	ASC (По возрастанию)

Таблица 5.6. Список связей для таблицы ROUTE

Связь для таблицы	Первичный ключ	Вторичный ключ
DESTINATION	Key_des	Key_des
TOUR	Key_tou	Key_tou

В табл. 5.7—5.9 содержится описание таблицы SCHEDULE.

Таблица 5.7. Список полей для таблицы SCHEDULE

Имя поля	Описание	Тип	Р	М
Key_sch	Key schedule (Уникальный ключ расписания)	Integer	Да	Да
Key_sea	Key season (Ключ сезона)	Integer	Нет	Да
Key_tou	Key tour (Ключ круиза)	Integer	Нет	Нет
TimeDepart_sch	Departure time (Время отправления)	Time	Нет	Да
ChBoxWorkDay_sch	Check box workday (Признак рабочего дня)	Numeric(1)	Нет	Нет

Таблица 5.8. Список индексов для таблицы *SCHEDULE*

Название индекса	P	F	U	C	Индексируемое поле	Сортировка
SCHEDULE_PK	Да	Нет	Да	Нет	Key_sch	ASC (По возрастанию)
sea_to_sch_FK	Нет	Да	Нет	Нет	Key_sea	ASC (По возрастанию)
tou_to_sch_FK	Нет	Да	Нет	Нет	Key_tou	ASC (По возрастанию)

Таблица 5.9. Список связей для таблицы *SCHEDULE*

Связь для таблицы	Первичный ключ	Вторичный ключ
SEASON	Key_sea	Key_sea
TOUR	Key_tou	Key_tou
TICKET	Key_sch	Key_sch

В табл. 5.10—5.12 приведено описание таблицы *SEASON*.

Таблица 5.10. Список полей для таблицы *SEASON*

Имя поля	Описание	Тип данных	P	M
Key_sea	Key season (Уникальный ключ сезона)	Integer	Да	Да
Begin_sea	Beginning of a season (Дата начало сезона)	Date	Нет	Да
End_sea	The ending of a season (Дата окончания сезона)	Date	Нет	Да

Таблица 5.11. Список индексов для таблицы *SEASON*

Название индекса	P	F	U	C	Индексируемое поле	Сортировка
SEASON_PK	Да	Нет	Да	Нет	Key_sea	ASC (По возрастанию)

Таблица 5.12. Список связей для таблицы *SEASON*

Связь для таблицы	Первичный ключ	Вторичный ключ
SCHEDULE	Key_sea	Key_sea

В табл. 5.13—5.15 содержится описание таблицы `TICKET`.

Таблица 5.13. Список полей для таблицы `TICKET`

Имя поля	Описание	Тип данных	P	M
<code>Key_tic</code>	Key ticket (Уникальный ключ билета)	<code>Integer</code>	Да	Да
<code>Key_tou</code>	Key tour (Ключ круиза)	<code>Integer</code>	Нет	Нет
<code>Key_tra</code>	Key traveler (Ключ путешественника)	<code>Integer</code>	Нет	Да
<code>Key_sch</code>	Key schedule (Ключ расписания)	<code>Integer</code>	Нет	Нет
<code>DateOrder_tic</code>	Date of the order (Дата заказа)	<code>Timestamp</code>	Нет	Да

Таблица 5.14. Список индексов для таблицы `TICKET`

Название индекса	P	F	U	C	Индексируемое поле	Сортировка
<code>TICKET_PK</code>	Да	Нет	Да	Нет	<code>Key_tic</code>	ASC (По возрастанию)
<code>tou_to_tic_FK</code>	Нет	Да	Нет	Нет	<code>Key_tou</code>	ASC (По возрастанию)
<code>tra_to_tic_FK</code>	Нет	Да	Нет	Нет	<code>Key_tra</code>	ASC (По возрастанию)
<code>sch_to_tic_FK</code>	Нет	Да	Нет	Нет	<code>Key_sch</code>	ASC (По возрастанию)

Таблица 5.15. Список связей для таблицы `TICKET`

Связь для таблицы	Первичный ключ	Вторичный ключ
<code>SCHEDULE</code>	<code>Key_sch</code>	<code>Key_sch</code>
<code>TOUR</code>	<code>Key_tou</code>	<code>Key_tou</code>
<code>TRAVELER</code>	<code>Key_tra</code>	<code>Key_tra</code>

В табл. 5.16—5.18 приведено описание таблицы `TOUR`.

Таблица 5.16. Список полей для таблицы `TOUR`

Имя поля	Описание	Тип данных	P	M
<code>Key_tou</code>	Key tour (Уникальный ключ круиза)	<code>Integer</code>	Да	Да

Таблица 5.16 (окончание)

Имя поля	Описание	Тип данных	P	M
Name_tou	Name tour (Наименование круиза)	Varchar(100)	Нет	Да
PointDepart_tou	Point of departure (Пункт отправления)	Varchar(100)	Нет	Да
NumLandingPlaces_tou	Number of landing places (Число посадочных мест)	Numeric(4,0)	Нет	Да
Time_tou	Time in tour, hours (Время в пути, в часах)	Numeric(4,2)	Нет	Да
CostAdult_tou	Cost for adults, \$ (Стоимость круиза для взрослых)	Numeric(6,2)	Нет	Да
CostChildren_tou	Cost for children, \$ (Стоимость круиза для детей)	Numeric(6,2)	Нет	Да
CostSenior_tou	Cost for senior citizens, \$ (Стоимость круиза для пожилых людей)	Numeric(6,2)	Нет	Да
Photo_tou	Photo tour (Фотография круиза)	Varchar(100)	Нет	Нет
Note_tou	Note tour (Примечание)	Long varchar	Нет	Нет

Таблица 5.17. Список индексов для таблицы *TOUR*

Название индекса	P	F	U	C	Индексируемое поле	Сортировка
TOUR_PK	Да	Нет	Да	Нет	Key_tou	ASC (По возрастанию)

Таблица 5.18. Список связей для таблицы *TOUR*

Связь для таблицы	Первичный ключ	Вторичный ключ
ROUTE	Key_tou	Key_tou
SCHEDULE	Key_tou	Key_tou
TICKET	Key_tou	Key_tou

В табл. 5.19—5.21 приведено описание таблицы *TRAVELER*.

Таблица 5.19. Список полей для таблицы TRAVELER

Имя поля	Описание	Тип данных	P	M
Key_tra	Key traveler (Уникальный ключ путешественника)	Integer	Да	Да
FirstName_tra	First name (Имя)	Varchar(20)	Нет	Да
MiddleInitial_tra	Middle initial (Инициал отчества)	Varchar(1)	Нет	Нет
LastName_tra	Last name (Фамилия)	Varchar(20)	Нет	Да
EMail_tra	E-Mail Address (Электронный адрес)	Varchar(50)	Нет	Да
Password_tra	Pssword traveler (Пароль)	Varchar(20)	Нет	Да
PhoneNum_tra	Phone number (Телефон)	Varchar(14)	Нет	Нет
Gender_tra	Gender (Пол)	Numeric(1)	Нет	Нет
Birthday_tra	Date of birth (Дата рождения)	Date	Нет	Нет

Таблица 5.20. Список индексов для таблицы TRAVELER

Название индекса	P	F	U	C	Индексируемое поле	Сортировка
TRAVELER_PK	Да	Нет	Да	Нет	Key_tra	ASC (По возрастанию)

Таблица 5.21. Список связей для таблицы TRAVELER

Связь для таблицы	Первичный ключ	Вторичный ключ
TICKET	Key_tra	Key_tra

Связи (References)

В табл. 5.22 содержится описание связей базы данных CRUISE.

Таблица 5.22. Список связей базы данных CRUISE

Имя связи	Код связи	Главное поле из таблицы	Подчиненное поле из таблицы
Des_to_rou	des_to_rou	DESTINATION	ROUTE
Sch_to_tic	sch_to_tic	SCHEDULE	TICKET
Sea_to_sch	sea_to_sch	SEASON	SCHEDULE

Таблица 5.22 (окончание)

Имя связи	Код связи	Главное поле из таблицы	Подчиненное поле из таблицы
Tou_to_rou	tou_to_rou	TOUR	ROUTE
Tou_to_sch	tou_to_sch	TOUR	SCHEDULE
Tou_to_tic	tou_to_tic	TOUR	TICKET
Tra_to_tic	tra_to_tic	TRAVELER	TICKET

Триггеры (Triggers)

Заметим, что триггер является мощным инструментом контроля за изменением данных в БД, а также помогает автоматизировать операции, которые должны выполняться в определенных случаях.

Триггеры — это предварительно определенное действие или последовательность действий, автоматически осуществляемых при выполнении операций обновления, добавления или удаления данных.

В табл. 5.23 приведен список триггеров базы данных CRUISE.

Таблица 5.23. Список триггеров для базы данных CRUISE

Таблица	Триггер	Определен пользователем
ROUTE	tib_route	Нет
SCHEDULE	tib_schedule	Нет
TICKET	tib_ticket	Нет

Далее представлены триггеры tib_route, tib_schedule и tib_ticket, сформированные автоматически после генерации физической модели базы данных.

Введем обозначения:

- ☐ полное наименование таблицы — ROUTE;
- ☐ имя таблицы, содержащей данный триггер, — ROUTE;
- ☐ наименование триггера — tib_route;
- ☐ тип триггера, определяющий характер действия BeforeInsertTrigger (Перед вставкой), — Default Trigger.

Листинг 5.1 содержит код триггера tib_route.

Листинг 5.1. Код триггера tib_route

```
% Before insert trigger "tib_route" for table "ROUTE"
create trigger tib_route before insert on ROUTE
```

```
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare insert_child_parent_exist_exception exception for SQLSTATE
'99991';
    declare found integer;

    % Parent "TOUR" must exist when inserting a child in "ROUTE"
    if (new_ins.Key_tou is not null) then
    begin
        set found = 0;
        select 1
        into found
        from dummy
        where exists (select 1
                        from TOUR
                        where Key_tou = new_ins.Key_tou);
        if found <> 1 then
            signal insert_child_parent_exist_exception
        end if;
    end
    end if;

    % Parent "DESTINATION" must exist when inserting a child in "ROUTE"
    if (new_ins.Key_des is not null) then
    begin
        set found = 0;
        select 1
        into found
        from dummy
        where exists (select 1
                        from DESTINATION
                        where Key_des = new_ins.Key_des);
        if found <> 1 then
            signal insert_child_parent_exist_exception
        end if;
    end
    end if;
exception
```



```

    if found <> 1 then
        signal insert_child_parent_exist_exception
    end if;
end
end if;

% Parent "TOUR" must exist when inserting a child in "SCHEDULE"
if (new_ins.Key_tou is not null) then
begin
    set found = 0;
    select 1
        into found
        from dummy
        where exists (select 1
                        from TOUR
                        where Key_tou = new_ins.Key_tou);
    if found <> 1 then
        signal insert_child_parent_exist_exception
    end if;
end
end if;
exception
when insert_child_parent_exist_exception then
    message 'Error: Trigger(tib_schedule) of table SCHEDULE';
    message '          Parent code must exist when inserting a child!';
    signal user_defined_exception;
when others then
    message 'Exception in before insert trigger(tib_schedule) of table
SCHEDULE';
    resignal;
end
/

```

Введем обозначения:

- ☐ полное наименование таблицы — `TICKET`;
- ☐ имя таблицы, содержащей данный триггер, — `TICKET`;
- ☐ наименование триггера — `tib_ticket`;
- ☐ тип триггера, определяющий характер действия `BeforeInsertTrigger` (Перед вставкой), — `Default Trigger`.


```
    if found <> 1 then
        signal insert_child_parent_exist_exception
    end if;
end
end if;

% Parent "SCHEDULE" must exist when inserting a child in "TICKET"
if (new_ins.Key_sch is not null) then
begin
    set found = 0;
    select 1
        into found
        from dummy
        where exists (select 1
                        from SCHEDULE
                        where Key_sch = new_ins.Key_sch);
    if found <> 1 then
        signal insert_child_parent_exist_exception
    end if;
end
end if;
exception
when insert_child_parent_exist_exception then
    message 'Error: Trigger(tib_ticket) of table TICKET';
    message '          Parent code must exist when inserting a child!';
    signal user_defined_exception;
when others then
    message 'Exception in before insert trigger(tib_ticket) of table
TICKET';
    resignal;
end
/
```

Дизайн проекта

Как вы помните, перед нами поставлена задача — разработка сайта для предоставления информации о круизах и экскурсиях. После ответа на ряд ключевых вопросов и создания базы данных для рассматриваемой задачи наступает время спроектировать непосредственно сам сайт, тщательно продумав его дизайн.

Web-дизайн — определенно творческий процесс: здесь вы можете проявить свои художественные способности, причем, как показывает практика, не обязательно иметь художественное образование. Сейчас существует достаточно много программных средств, позволяющих работать с графикой, обрабатывать фотографии использовать всевозможные шрифты. Однако чувство меры и вкус никогда не помешают. И если вам не довелось прослушать курс лекций по Web-дизайну, то позвольте дать несколько советов:

- ❑ Web-страница должна идентично отображаться в Microsoft Internet Explorer и Netscape Navigator;
- ❑ в силу разных версий как для Microsoft Internet Explorer, так и для Netscape Navigator включайте проверку версии текущего браузера для формирования соответствующих страниц;
- ❑ избегайте размещения на первой странице большого количества графических объектов, существенно затрудняющих загрузку документа;
- ❑ не злоупотребляйте шрифтами, двух-трех различных шрифтов вполне достаточно;
- ❑ используйте гармонирующие между собой цветовые схемы, если конечно вы не разрабатываете "модерновый" сайт, предназначенный для специальных целей;
- ❑ избегайте размещения копий чужих фотографий;
- ❑ продумывайте удобное средство навигации, чтобы ваш сайт не оказался лабиринтом для потенциальных посетителей;
- ❑ прежде чем "выставлять" сайт, продемонстрируйте результат своего труда знакомым, вполне возможно, что вам помогут найти незамеченные ранее ошибки.

В разных источниках можно прочитать, что главный критерий, на который следует ориентироваться в процессе разработки страниц, — это удобство конечного пользователя. И с этим трудно не согласиться.

Одним из первых шагов является набросок "карты сайта". Это графическая схема, показывающая путь продвижения посетителя по сайту. Ваша карта должна включать в себя каждую его страницу и то, в какой зависимости они находятся относительно других страниц.

Для нашего случая определим шесть основных пунктов, характеризующих костяк сайта:

- ❑ Tour — детальная информация по каждому круизу;
- ❑ Scheduler — расписание круизов (экскурсий);
- ❑ Guidance — всевозможные наставления с дополнительной информацией;
- ❑ Buy Tickets — покупка билетов;

- City Map — карта города;
- Contact Us — контакты для связи.

Далее определим, следует ли нам использовать так называемую заглавную страницу (splash page). Заглавная страница (или заставка) чаще всего применяется, если к потенциальному посетителю существует ряд вопросов, или возникает необходимость проинформировать посетителя о чем-либо, например, выбор языка или кодировки страниц, демонстрация Flash-анимации, предупреждение о том или ином соглашении при использовании данных текущего ресурса, демонстрация счетчика посещений, показ рейтинговых форм сайта и т. д.

Единственное, для чего следует использовать заглавную страницу в нашем случае, — это выбор языка просмотра страницы, поскольку среди путешественников достаточно часто встречаются иностранные туристы, предпочитающие читать на родном языке. Но в данном случае есть предложение перейти к разработке английской версии Web-приложения. (Разработку заглавной страницы разберем при изучении cookies в *главе 7*.)

Любая Web-страница содержит определенный набор стандартных элементов, таких как заголовок, логотип, элементы навигации, основная область страницы, подзаголовок и т. д. Компоновка этих элементов, проектирование их взаимного расположения и составляют одну из главных задач Web-мастера.

Еще одним немаловажным шагом является выбор используемых технологий при кодировании страниц.

Неспроста мы уделили значительное время разработке базы данных, т. к. некоторые наши страницы в силу своей специфики будет проще формировать "на лету", используя саму базу данных, например для составления расписания круизов.

Для упрощения работы с набором стилей Web-проекта как при создании, так и при дальнейшей его поддержке предполагаем использование технологии каскадных таблиц стилей (Cascading Style Sheets, CSS).

Также в отдельных случаях планируем использование Dynamic HTML и JavaScript. Мы уже не оговариваем, что ключевым моментом при кодировании документов будет язык CFML.

После определенных усилий по части выбора собственного стиля и компоновке основных элементов страницы, мы пришли к результату, отраженному на рис. 5.4.

Следуя изложенным ранее советам, хочется отметить, что фотография с видом статуи свободы — также результат собственного творчества.

В итоге последовательность работ по проектированию Web-приложения сводится к следующим действиям:

- постановка задачи;
- определение бизнес-правил;



Рис. 5.4. Интерфейс сайта "My-Line Cruises"

- ☐ разработка базы данных (при необходимости);
- ☐ определение структуры сайта;
- ☐ выбор используемых технологий;
- ☐ стилизация, разработка дизайна и компоновка;
- ☐ подготовка текстовых материалов;
- ☐ отладка Web-страниц;
- ☐ проверка идентичности отображения Web-страниц с различным экран-ным разрешением и в различных браузерах.

Имея достаточный опыт работы по проектированию приложений, отметим, что, чуть ли ни каждый этап разработки Web-приложения следует согласо-вывать с заказчиком.

В *приложении 1* представлены коды основных файлов в рамках проекта "My-Line Cruises".



Первая страница с использованием CFML

Вы можете создавать CFML-страницы при помощи любого HTML-редактора или даже обычного Notepad (Блокнота). Однако рекомендуем использовать ColdFusion Studio. ColdFusion Markup Language (CFML) — это язык разметки, синтаксис которого подобен HTML. Поэтому для Web-программистов CFML может являться интуитивно понятным средством разработки.

В отличие от HTML, который определяет отображение и форматирование элементов на компьютере клиента, CFML опознает определенные действия, выполняемые сервером ColdFusion. Открытый и расширяемый язык CFML предоставляет более 75 серверных тегов, 247 функций и 1000 компонентов третьих фирм. Кроме того, ColdFusion легко дополняется с помощью приложений на Java и C++.

Пишем код CFML-страницы

Рассмотрим пример создания страницы с использованием CFML. Итак:

1. Откройте ColdFusion Studio.
2. Выберите команду **File | New (Файл | Создать)** и определите новый документ как **Default Template (Шаблон по умолчанию)**.
3. Отредактируйте новый файл так, чтобы в результате у вас получился код, представленный в листинге 6.1.

Листинг 6.1. Текст нового файла

```
<HTML>
<HEAD>
  <TITLE>Welcome</TITLE>
</HEAD>
<BODY>
```

```
<STRONG>Welcome!</STRONG>
<!-- Установка переменной FirstName --->
<CFSET FirstName="Anjuta">
<!-- Вывод на экран --->
<CFOUTPUT>
    Dear Ms. #FirstName#, thank you for your interest in CFML.
</CFOUTPUT>
</BODY>
</HTML>
```

Сохранение CFML-страниц

ColdFusion поддерживает следующие расширения: cfm и cfml. По умолчанию ColdFusion Studio сохраняет новые файлы с расширением cfm.

Для сохранения нашего примера выполните следующие действия:

- 1. Выберите команду **File | Save (Файл | Сохранить)**.
- 2. Сохраните свою страницу с именем `Welcome.cfm` в корневом каталоге Web-сервера.

Например, если в корневом каталоге Web-сервера определен каталог my-app, тогда для:

- ☐ Windows NT IIS выберите `c://inetpub/wwwroot/my-app`;
- ☐ Windows NT Apache укажите `c://apache/htdocs/my-app`;
- ☐ UNIX задайте `<mywebserverdocroot>/my-app`.

Просмотр CFML-страниц

Для просмотра страницы необходимо открыть Web-браузер и в поле URL-адреса набрать `http://127.0.0.1/my-app/welcome.cfm`, где 127.0.0.1 — адрес локальной машины — localhost.

Сравните по табл. 6.1 код, созданный первоначально, и код, возвращаемый Web-сервером.

Таблица 6.1. Первоначальный и возвращенный Web-сервером коды

Первоисточник, созданный в ColdFusion	Просмотр в виде HTML на компьютере клиента
<HTML>	<HTML>
<HEAD>	<HEAD>
<TITLE>Welcome</TITLE>	<TITLE>Welcome</TITLE>

Таблица 6.1 (окончание)

Первоисточник, созданный в ColdFusion	Просмотр в виде HTML на компьютере клиента
<pre></HEAD> <BODY> Welcome! <!-- Установка переменной FirstName ---> <CFSET FirstName="Anjuta"> <!-- Вывод на экран ---> <CFOUTPUT> Dear Ms. #FirstName#, thank you for your interest in CFML. </CFOUTPUT> </BODY> </HTML></pre>	<pre></HEAD> <BODY> Wel come! Dear Ms. Anjuta, thank you for your interest in CFML. </BODY> </HTML></pre>

Также просмотр CFML-страниц можно осуществлять непосредственно из ColdFusion Studio, предварительно произведя следующие установки:

1. Выберите команду **Options | Settings** (Настройка | Параметры) и из левого списка выделите раздел **Browse** (Обзор) — рис. 6.1.
2. Нажмите кнопку **Development Mappings** (Подготовка псевдонимов), расположенную в центре окна.
3. На вкладке **Mappings** (Псевдонимы) нажмите кнопку **Add** (Добавить) и в открывшемся диалоговом окне укажите параметры: **Studio Path** (Месторасположение Web-сервера с точки зрения Studio), **CF Server Path** (Месторасположение Web-сервера с точки зрения CF Server) и **Browser Path** (Виртуальный путь при просмотре с использованием браузера), в зависимости от Web-сервера. При размещении CF Studio и CF Server на одной машине, значения полей **Studio Path** и **CF Server Path** идентичны.
4. Нажмите кнопку **ОК**.
5. Далее подтвердите свой выбор щелчком по кнопке **ОК** уже в окне **Remote Development Settings** (Настройка удаленных псевдонимов) — рис. 6.2.

Обратите внимание, что таким образом можно создавать сразу несколько логических псевдонимов (mappings) для ваших Web-разработок. На рис. 6.2 установлено два логических псевдонима, для разрабатываемого нами приложения под управлением Apache Web Server и уже заранее созданного проекта для MS Internet Information Server.

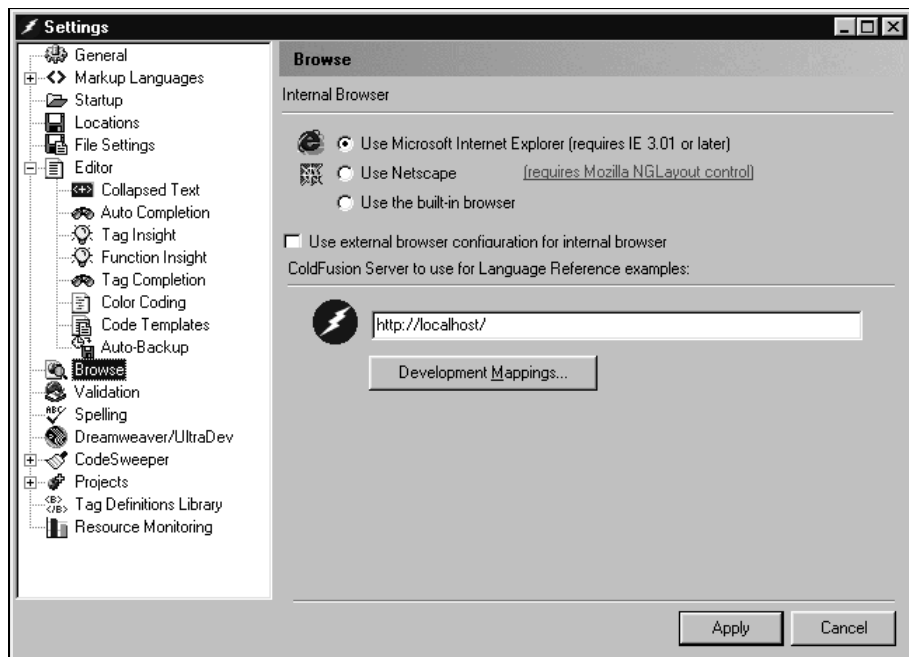


Рис. 6.1. Диалоговое окно **Settings**, раздел **Browse**

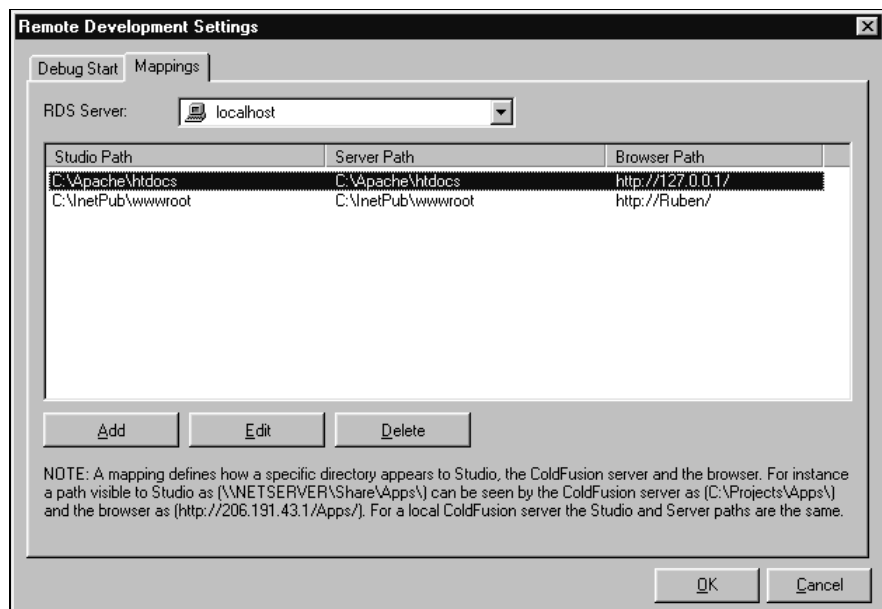


Рис. 6.2. Диалоговое окно **Remote Development Settings**, вкладка **Mappings**

В дальнейшем для просмотра результата написанного нами кода остается воспользоваться панелью режима работы с документами и перейти из режима редактирования **Edit** (Редактор) в режим просмотра **Browse** (Просмотр) — рис. 6.3.

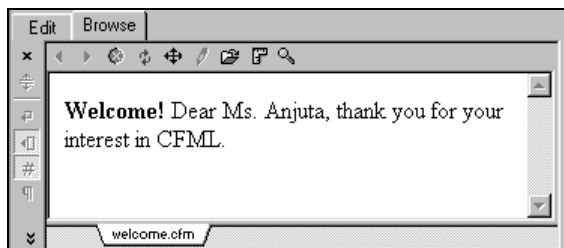


Рис. 6.3. Результат отображения страницы welcome.cfm

Анализ кода

В рассматриваемом достаточно простом примере мы используем тег `<CFSET>` для определения переменной `FirstName`. После чего внутри границ тега `<CFOUTPUT>` помещаем текст, включающий значение данной переменной. Для вывода значения переменной заключаем ее между символами `#`.

Таблица 6.2. Описание кода страницы welcome.cfm

Код	Описание
<code><!--- Установка переменной FirstName ---></code>	CFML-комментарий, который не отображается в HTML-странице
<code><CFSET FirstName="Anjuta"></code>	Создаем переменную с именем <code>FirstName</code> и задаем значение, равное <code>"Anjuta"</code>
<code><!--- Вывод на экран ---></code>	CFML-комментарий, который не отображается в HTML-странице
<code><CFOUTPUT></code> <code>Dear Ms. #FirstName#, thank you for your interest in CFML.</code> <code></CFOUTPUT></code>	Выводим на экран содержимое тега <code><CFOUTPUT></code> , в нашем случае текст <code>"Dear Ms."</code> ("Уважаемая"), затем помещаем значение переменной <code>FirstName</code> и далее фразу <code>", thank you for your interest in CFML."</code> ("спасибо за оказанное внимание к CFML")

Использование переменных — достаточно удобное средство для создания гибких или, скажем, динамических страниц, когда содержимое HTML-документа может изменяться в зависимости от состояния того или иного элемента.

Усложнение кода

Предположим, нам необходимо изначально определить имя и пол пользователя и в зависимости от этого поместить сообщение приветствия. Для этой цели мы будем сочетать форму стандарта CGI и CFML.

Создадим новую CFML-страницу с именем `welcome_flex.cfm` и напишем соответствующий код (листинг 6.2).

Листинг 6.2. Код страницы `welcome_flex.cfm`

```
<HTML>

<HEAD>

  <META name="Author" content="Ruben Akhayan">
  <META http-equiv="Description" content="The advanced code">
  <META http-equiv="Keywords" content="ColdFusion, CFML, Book">
  <TITLE>Welcome</TITLE>

</HEAD>

<BODY>

  <!-- Проверка на наличие переменных --->
  <CFIF isDefined("Form.FirstName") AND isDefined("Form.Gender")>
    <STRONG>Welcome!</STRONG>
    <!-- Установка переменной Appeal --->
    <CFIF #Form.Gender# eq "f">
      <CFSET Appeal="Ms.">
    <CFELSE>
      <CFSET Appeal="Mr.">
    </CFIF>
    <!-- Вывод на экран --->
    <CFOUTPUT>
      Dear #Appeal# #Form.FirstName#, thank you for your interest
      in CFML.
    </CFOUTPUT>
  <CFELSE>
    <!-- Скрипт для контроля ввода данных --->
    <script>
      function IsEmpty(data){
        if(data.length==0) return true;
```

```

        else return false;
    }
    function IsVerification(){
        if(IsEmpty(document.form1.FirstName.value)){
            alert("Please, enter your Name!");
            return false;
        }
    }
</script>
<!-- Форма для ввода данных --->
<FORM action='welcome_flex.cfm' method='post' name='form1'
        onSubmit='return IsVerification();'>
<TABLE border='0' cellpadding='5' cellspacing='0'>
    <TR><TD align="center" colspan="2">
        Please, fill in the data on itself!</TD></TR>
    <TR><TD colspan="2">&nbsp;</TD></TR>
    <TR><TD align="right">First Name:</TD>
        <TD><input name='FirstName' size='17'></TD></TR>
    <TR><TD align="right">Gender:</TD>
        <TD><select name='Gender'>
            <option value="m">male
            <option value="f">female
        </select>
        </TD></TR>
    <TR><TD colspan="2">&nbsp;</TD></TR>
    <TR><TD align="right">
        <input type='submit' value='Send'></td><td>
        <input type='reset' value='Clear'></td></tr>
</TABLE>
</FORM>
</CFIF>
</BODY>
</HTML>

```

Теперь разберем, что происходит на этой странице, уже не рассматривая CFML-комментарии и классические HTML-теги (табл. 6.3).

Таблица 6.3. Описание кода страницы *welcome_flex.cfm*

Код	Описание
<pre><CFIF isDefined("Form.FirstName") AND isDefined("Form.Gender")></pre>	Включаем тег <code><CFIF></code> условного оператора для проверки наличия переменных <code>FirstName</code> и <code>Gender</code> . Если переменные определены, тогда в HTML-страницу помещается код, расположенный ниже вплоть до раздела <code><CFELSE></code>
<pre><CFIF #Form.Gender# eq "f"> <CFSET Appeal="Ms."> <CFELSE> <CFSET Appeal="Mr."> </CFIF></pre>	Создаем переменную обращения к пользователю с именем <code>Appeal</code> и задаем значение в зависимости от пола
<pre><CFOUTPUT> Dear #Appeal# #Form.FirstName#, thank you for your interest in CFML. </CFOUTPUT></pre>	Выводим на экран сообщение приветствия, включая значения переменных <code>Appeal</code> и <code>FirstName</code>
<pre><CFELSE></pre>	Раздел тега условного оператора <code><CFIF></code> , "Иначе". Если не выполняется условие оператора, тогда в HTML-страницу помещается код, расположенный ниже и до раздела окончания тега оператора <code><CFIF></code>
<pre><script> function IsEmpty(data){ if(data.length==0)return true; else return false; } function IsVerification(){ if(IsEmpty(document.form1.FirstName.value)){ alert("Please, enter your Name!"); return false; } } </script></pre>	Формируем сценарий для контроля ввода данных в регистрационной форме. В нашем случае предполагаем, что поле <code>FirstName</code> является обязательным, и нельзя допустить пустого значения. При несоблюдении данного правила выводим сообщение с текстом "Please, enter your Name!" (Пожалуйста, введите Ваше имя!)

Таблица 6.3 (окончание)

Код	Описание
<pre> <FORM action='welcome_flex.cfm' method='post' name='form1' onSubmit='return IsVerification();'> <TABLE border='0' cellspacing='5' cellpadding='0'> <TR><TD align="center" colspan="2"> Please, fill in the data on itself!</TD></TR> <TR><TD colspan="2"> &nbsp; </TD></TR> <TR><TD align="right"> First Name:</TD> <TD><input name='FirstName' size='17'></TD></TR> <TR><TD align="right"> Gender:</TD> <TD><select name='Gender'> <option value="m">male <option value="f">female </select> </TD></TR> <TR><TD colspan="2"> &nbsp; </TD></TR> <TR><TD align="right"> <input type='submit' value='Send'></td><td> <input type='reset' value='Clear'></TD></TR> </TABLE> </FORM> </pre>	Предлагаем пользователю ввести данные о себе — имя и пол, используя форму стандарта CGI. Надо заметить, что при выполнении команды отправки сведений мы обращаемся к этой же странице <code>welcome_flex.cfm</code>. Однако включенный в страницу условный оператор, проверяющий наличие переменных <code>FirstName</code> и <code>Gender</code>, избавит пользователя от повторной регистрации. В чем, собственно говоря, и заключается изюминка (рис. 6.4)
<pre> </CFIF> </pre>	Конец тега <code><CFIF></code> условного оператора

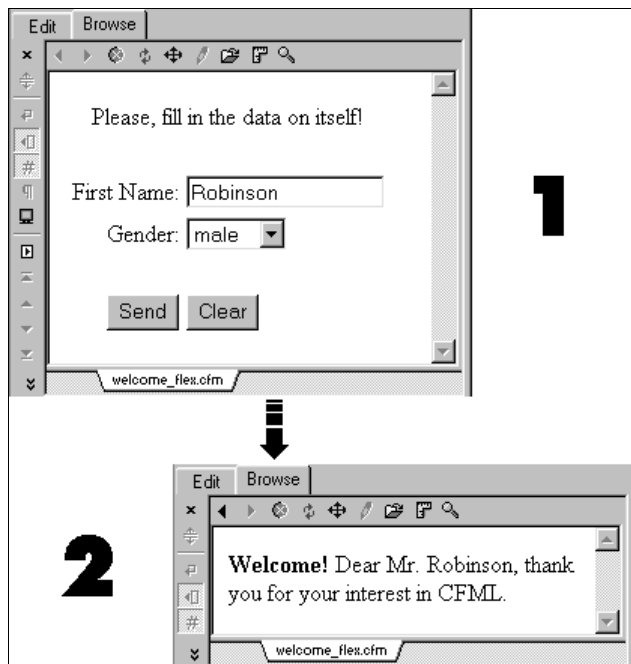


Рис. 6.4. Результат отображения страницы welcome_flex.cfm после регистрации

Рассмотрим код страницы welcome_flex.cfm в виде HTML при первом посещении (листинг 6.3) и после регистрации (листинг 6.4).

Листинг 6.3. Код страницы welcome_flex.cfm при первом посещении

```
<HTML>
<HEAD>
  <META name="Author" content="Ruben Akhayan">
  <META http-equiv="Description" content="The advanced code">
  <META http-equiv="Keywords" content="ColdFusion, CFML, Book">
  <TITLE>Welcome</TITLE>
</HEAD>
<BODY>
  <script>
    function IsEmpty(data){
      if(data.length==0) return true;
      else return false;
    }
  </script>
  <div>
    <div>Please, fill in the data on itself!</div>
    <div>
      First Name: <input type="text" value="Robinson"/>
      Gender: <input type="text" value="male"/>
      <input type="button" value="Send"/>
      <input type="button" value="Clear"/>
    </div>
  </div>
</BODY>
</HTML>
```



```

function IsVerification(){
    if(IsEmpty(document.form1.FirstName.value)){
        alert("Please, enter your Name!");
        return false;
    }
}
</script>
<FORM action='welcome_flex.cfm' method='post' name='form1'
    onSubmit='return IsVerification();'>
<TABLE border='0' cellspacing='5' cellpadding='0'>
    <TR><TD align="center" colspan="2">
        Please, fill in the data on itself!</TD></TR>
    <TR><TD colspan="2">&nbsp;</td></tr>
    <TR><TD align="right">First Name:</td>
    <TD><input name='FirstName' size='17'></TD></TR>
    <TR><TD align="right">Gender:</TD>
    <TD><select name='Gender'>
        <option value="m">male
        <option value="f">female
    </select>
    </TD></TR>
    <TR><TD colspan="2">&nbsp;</TD></TR>
    <TR><TD align="right">
        <input type='submit' value='Send'></TD><TD>
        <input type='reset' value='Clear'></TD></TR>
</TABLE>
</FORM>
</BODY>
</HTML>

```

Листинг 6.4. Код страницы welcome_flex.cfm после регистрации

```

<HTML>
<HEAD>
    <META name="Author" content="Ruben Akhayan">
    <META http-equiv="Description" content="The advanced code">
    <META http-equiv="Keywords" content="ColdFusion, CFML, Book">
    <TITLE>Welcome</TITLE>
</HEAD>

```

```
<BODY>
  <STRONG>Welcome!</STRONG>
  Dear Mr. Robinson, thank you for your interest in CFML.
</BODY>
</HTML>
```

Поставленную задачу можно также решить, используя две CFML-страницы. Для чего содержимое страницы `welcome_flex.cfm` разобьем на два файла: `welcome_form.cfm` (листинг 6.5) и `welcome_go.cfm` (листинг 6.6), где файл `welcome_form.cfm` будет представлять собой непосредственно форму ввода данных, а файл `welcome_go.cfm` — приемник данных с последующим выводом на экран сообщения приветствия.

Листинг 6.5. Код страницы `welcome_form.cfm`

```
<HTML>
<HEAD>
  <META name="Author" content="Ruben Akhayan">
  <TITLE>Welcome</TITLE>
  <!-- Сценарий для контроля ввода данных -->
  <script>
    function IsEmpty(data){
      if(data.length==0)return true;
      else return false;
    }
    function IsVerification(){
      if(IsEmpty(document.form1.FirstName.value)){
        alert("Please, enter your Name!");
        return false;
      }
    }
  </script>
</HEAD>
<BODY>
  <!-- Форма для ввода данных -->
  <FORM action='welcome_go.cfm' method='post' name='form1'
    onsubmit='return IsVerification();'>
    <TABLE border='0' cellspacing='5' cellpadding='0'>
      <TR><TD align="center" colspan="2">
        Please, fill in the data on itself!</TD></TR>
```

```

<TR><TD colspan="2">&nbsp;</td></tr>
<TR><TD align="right">First Name:</td>
<TD><input name='FirstName' size='17'></TD></TR>
<TR><TD align="right">Gender:</TD>
<TD><select name='Gender'>
    <option value="m">male
    <option value="f">female
</select>
</TD></TR>
<TR><TD colspan="2">&nbsp;</TD></TR>
<TR><TD align="right">
    <input type='submit' value='Send'></td><td>
    <input type='reset' value='Clear'></td></tr>
</TABLE>
</FORM>
</BODY>
</HTML>

```

Листинг 6.6. Код страницы welcome_go.cfm

```

<HTML>
<HEAD>
    <META name="Author" content="Ruben Akhayan">
    <TITLE>Welcome</TITLE>
</HEAD>
<BODY>
    <!-- Проверка наличия переменных --->
    <CFIF isDefined("Form.FirstName") AND isDefined("Form.Gender")>
        <STRONG>Welcome!</STRONG>
    <!-- Установка переменной Appeal --->
    <CFIF #Form.Gender# eq "f">
        <CFSET Appeal="Ms.">
    <CFELSE>
        <CFSET Appeal="Mr.">
    </CFIF>
    <!-- Вывод на экран --->
    <CFOUTPUT>
        Dear #Appeal# #Form.FirstName#, thank you for your interest
        in CFML.
    </CFOUTPUT>

```

```
</CFOUTPUT>
<CFELSE>
  <!--- Данные не определены, отправляем на форму ввода данных --->
  <META HTTP-EQUIV="Refresh" CONTENT="0;URL=welcome_form.cfm">
</CFIF>
</BODY>
</HTML>
```

В качестве примечания отметим, что страницу `welcome_form.cfm` можно смело преобразовать в HTML-формат, предварительно убрав комментарии CF, например, в `welcome_form.html`. Только в этом случае надо не забыть изменить ссылку на странице `welcome_go.cfm`, с `welcome_form.cfm` на `welcome_form.html`. Дело в том, что в коде данной страницы установлен переход на форму ввода данных при условии их отсутствия, что достаточно удобно для избежания ошибок и соблюдения следующего условия — регистрационная карта должна быть заполнена.

Резюме

Подведем итог. В данной главе мы познакомились и использованием CFML-тегов `<CFSET>`, `<CFOUTPUT>`, `<CFIF>/<CFELSE>`, функцией `IsDefined` и оператором `eq`.



Cookies в CFML

Cookies, так называемое "печенье" или, как еще иногда упоминают в литературе, ячейки, позволяет хранить информацию на стороне клиента для последующего использования. Это может быть информация о клиенте, воспользовавшемся услугами сайта, о содержимом "покупательской корзины" в электронном магазине, о действиях, произведенных клиентом по отношению к расположенным на странице элементам формы.

Cookie является решением одной из проблем протокола HTTP (HyperText Transfer Protocol, протокол передачи гипертекста), которая заключается в непостоянстве соединения между клиентом и сервером. Включение cookie в HTTP-протокол способствовало частичному решению этой проблемы. То есть транзакция завершается после того, как браузер сделал запрос, а сервер выдал соответствующий ответ. Сразу после этого сервер "забывает" о пользователе и каждый следующий запрос того же пользователя считает переданным новым пользователем.

С помощью cookie можно эмулировать сессию по HTTP-протоколу. Принцип эмуляции сессии таков: на первом запросе выдается соответствующее значение cookie, а при каждом последующем запросе это значение читается из переменной окружения HTTP_COOKIE и соответствующим образом обрабатывается.

Cookie — это небольшая порция текстовой информации, которую сервер передает браузеру. Браузер будет хранить эту информацию и передавать ее серверу с каждым запросом как часть HTTP-заголовка. Некоторые значения cookie могут храниться только в течение одной сессии и удаляются после закрытия браузера. Другие, настроенные на определенный период времени, записываются в файл. Обычно этот файл называется cookies.txt и размещается в рабочем каталоге установленного на компьютере браузера.

Тег <CFCOOKIE>

Синтаксис:

```
<CFCOOKIE NAME="cookie_name"  
          VALUE="text"
```

```
EXPIRES="period"
SECURE="Yes | No"
PATH="urls"
DOMAIN=".domain">
```

В табл. 7.1 представлено описание атрибутов тега <CFCOOKIE>.

Таблица 7.1. Описание атрибутов тега <CFCOOKIE>

Атрибут	Описание
NAME	Название переменной cookie. Обязательный атрибут
VALUE	Значение переменной cookie. Необязательный атрибут
EXPIRES	Планирует жизненный цикл переменной cookie. Может быть определен как дата (например, 12/31/2001), число дней (например, 10), <i>Now</i> (сейчас) или <i>Never</i> (никогда). Использование значения <i>Now</i> удаляет cookie из клиентского браузера. Является необязательным атрибутом и если не указывается, то cookie хранится в течение одного сеанса, до закрытия браузера
SECURE	Указывает, что переменная должна быть передана по надежному каналу. Если этот атрибут имеет значение <i>Yes</i> , то информация cookie пересылается только через HTTPS (HTTP с использованием SSL, Secure Socket Level) в защищенном режиме. Иначе информация пересылается обычным способом. В случае использования защищенного режима и отсутствия на клиентском браузере SSL, cookie не будет создан. Необязательный атрибут
PATH	Этот атрибут определяет подмножество документов, для которых действительно значение cookie. Например, указание <code>path="/my"</code> приведет к тому, что значение cookie будет действительно для множества документов в каталогах <code>/my-app1/</code> , <code>/my-app2/</code> и файлов в текущем каталоге с именами типа <code>myfile1.html</code> и <code>myfile2.shtml</code> . Для того чтобы cookie отсылались при каждом запросе к серверу, необходимо указать корневой каталог сервера, например, <code>path="/"</code> . Для перечисления нескольких значений необходимо использовать разделитель ; (точку с запятой). Необязательный атрибут
DOMAIN	Домен, для которого значение cookie действительно. Указанное значение должно всегда начинаться с точки, например, <code>domain=".Akhayan.com"</code> . Для перечисления нескольких значений необходимо использовать разделитель ; (точку с запятой). При использовании атрибута <code>PATH</code> необходимо указывать и атрибут <code>DOMAIN</code>

Cookies, написанные с использованием тега <CFCOOKIE>, не сохраняются в файле cookies.txt до конца сессии браузера. Пока браузер открыт, cookie

хранится в памяти. Если атрибут `EXPIRES` не указан, cookie не записывается в файл `cookies.txt` и существует до завершения сессии браузера.

Если cookie принимает новое значение при имеющемся уже в браузере cookie с совпадающими атрибутами `NAME`, `DOMAIN` и `PATH`, то старое значение заменяется новым. В остальных случаях новые значения cookie добавляются к старым.

Использование атрибута `EXPIRES` не гарантирует сохранность cookie в течение заданного периода времени, поскольку клиент (браузер) может удалить запись из-за нехватки выделенного места или по каким-либо другим причинам.

Клиент (браузер) имеет следующие ограничения для cookies:

- ☐ всего может храниться до 300 значений cookies;
- ☐ каждый cookie не может превышать 4 Кбайт;
- ☐ с одного сервера или домена может храниться до 20 значений cookie.

Если ограничение 300 или 20 превышает, то удаляется первая по времени запись. При превышении лимита объема в 4 Кбайт корректность значения cookie страдает — отсекается кусок записи (с начала этой записи), равный превышению объема.

Пример использования тега `<CFCOOKIE>`

Предположим, нам необходимо создать первую страницу сайта, предоставляющего информацию о круизах из Нью-Йорка, с предварительным запросом о национальном языке просмотра. При этом по согласованию с пользователем, в дальнейшем чтобы больше не озадачивать его подобными вопросами, в течение недели необходимо сразу загружать информацию на родном ему языке. Более того, немного усложним задачу: при загрузке выбранной пользователем страницы необходимо открывать окно с дополнительной информацией в течение всего времени до обусловленной даты. В качестве дополнительной информации может быть расписание сезонных рейсов, имеющее силу до определенного момента, или реклама с определенными сроками действия, или еще что-либо.

Решение данной задачи может быть следующим. Во-первых, нужно решить, сколько мы будем использовать cookies, а во-вторых, каким будет жизненный цикл для каждого cookie. Нам необходимо два cookies: первый будет определять выбранный язык, а второй — загрузку окна с дополнительной информацией. Для первого cookie, назовем его `language`, исходя из задачи, жизненный цикл укажем в семь дней. Для второго cookie, назовем его `commercial`, жизненный цикл зададим до конца 2001 года.

Следовательно, определение наших "пирожков" будет выглядеть следующим образом:

```
<CFCookie NAME="language"
          VALUE="#language#"
          EXPIRES="7">
```

где #language# — значение переменной.

```
<CFCookie NAME="commercial"
          VALUE="1"
          EXPIRES="12/31/2001">
```

Заметим, что "пирожок" commercial, будет "съеден" в ночь с 30 на 31 декабря 2001 года, а 31 числа от него уже ничего не останется, если только до этого момента он не будет "проглочен" клиентом браузера.

Для того чтобы прочитать значение cookie, в CFML используется конструкция Cookie.Name, где Name — наименование cookie. Например:

```
<CFIF isDefined("Cookie.language")>
  <CFSET language=#Cookie.language#>
</CFIF>
```

В этом коде проверяем наличие cookie с названием language, и если такой существует, создаем переменную с подобным именем, определяя ей значение в соответствии с Cookie.language.

Далее продемонстрируем полный код страницы my_line.cfm, позволяющей решить поставленную выше задачу (листинг 7.1).

Листинг 7.1. Код страницы my_line.cfm

```
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html;
    charset=windows-1251">
  <META name="Author" content="Ruben Akhayan">
  <META http-equiv="Description" content="Cookies">
  <TITLE>My-Line Cruises</TITLE>
</HEAD>
  <style type="text/css">
    <!--
      .clRemLang {
        padding : 0px;
        font-weight : normal;
```



```
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
color : #000000;
}
.clPrompt {
padding : 0px;
font-weight : bolder;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 17px;
color : #666699;
}
#prompt {position:absolute; left:10; top:51; width:400;
height:10;margin:0;z-index:1;}
-->
</style>
<script>
var ns = (document.layers)? true:false
var ie = (document.all)? true:false

function layerWrite(id,nestref,text) {
  if (ns) {
    var lyr = (nestref)? eval('document.'+nestref+'.document.'+id+
      '.document') : document.layers[id].document
    lyr.open()
    lyr.write(text)
    lyr.close()
  }
  else if (ie) document.all[id].innerHTML = text
}

function myLink(lang) {
  linecommand = "window.location.href = 'my_line.cfm?language="+
    lang+"&remlang="+document.form1.RemLang.value+"'";
  eval(linecommand);
}

function chbRemLang(value) {
  if (value=='Yes') {
    document.form1.RemLang.value = 'No';
```

```

    } else {
        document.form1.RemLang.value = 'Yes';
    }
}
</script>
<BODY>
<!-- Проверка наличия переменных --->
<CFIF isDefined("Cookie.language") OR isDefined("language")>
    <script>document.bgColor="#666699";</script>

    <CFIF isDefined("Cookie.language")>
        <CFSET language=#Cookie.language#>
    </CFIF>

    <CFIF isDefined("remlang")>
        <CFIF remlang eq "Yes">
            <!-- Устанавливаем Cookie-"language" --->
            <CFCOOKIE NAME="language"
                VALUE="#language#"
                EXPIRES="7">
        </CFIF>
    </CFIF>

<TABLE border='0' cellspacing='9' cellpadding='0'>
    <TR><TD align="center" colspan="2">
        <FONT size="5" color="White" face="Futura XBlk BT">
            MY-LINE CRUISES
        </FONT></TD></TR>
    <TR><TD bgcolor="White" height=1 colspan="2"></td></tr>
    <TR><TD></td>
    <TR><TD>
        <FONT size="2" color="White" face="Arial">
        <CFSWITCH EXPRESSION=#language#>
        <CFCASE VALUE="rus">
            <br>
            <br>
            (My Line).<br>
            ...

```

```
</CFCASE>
<CFCASE VALUE="eng">
    Discover why over 50 million people<br>
    have chosen My Line as the best way to see New York.<br>
    ...
</CFCASE>
<CFCASE VALUE="ger">
    Wie bereits mehr als 50 Millionen Besucher vir Ihnen,
    erleben und<br>
    entdecken auch Sie New York bei einer Fahrt auf ainem der
    My Line Schiffe.<br>
    ...
</CFCASE>
<CFCASE VALUE="spa">
    Unase a mas de 50 millones de visitantes que han
    palpado la<br>
    emocion de descubrir a Nueva York a bordo de un barco
    de My Line.<br>
    ...
</CFCASE>
<CFDEFAULTCASE>
    ...
</CFDEFAULTCASE>
</CFSWITCH>
</FONT>
</TD></TR>
</table>

<CFIF isDefined("Cookie.commercial") is False>
    <!-- Устанавливаем Cookie-"commercial" --->
    <CFCOOKIE NAME="commercial"
        VALUE="1"
        EXPIRES="12/31/2001">
<CFELSE>
    <CFIF #Cookie.commercial# eq 1>
        <script>
            Variable = window.open("commercial.cfm", "windowname",
                "top=10, left=530, width=250, height=490" );
```

```
</script>
</CFIF>
</CFIF>

<CFELSE> <!-- Cookie.language не определен -->
<DIV id="prompt">
<TABLE border='0' cellspacing='0' cellpadding='0'>
<TR><TD width='400' class='clPrompt' align='right'>
New York's BEST<br>Sightseeing and<br>Live
Music<br>Cruises
<TD></TR></TABLE>
</DIV>

<FORM method='post' name='form1'>
<TABLE border='0' cellspacing='2' cellpadding='0'>
<TR><TD colspan="2" width="400" align="center" bgcolor="#666699">
<FONT size="5" color="White" face="Futura XBlk BT">
MY-LINE CRUISES
</FONT></TD></TR>
<TR><TD colspan="2" bgcolor="#666699" height=1></TD></TR>
<TR><TD colspan="2" valign="bottom" align="center"
height="342">
<IMG src="pic1.gif" alt="" border="0" usemap="#MyMap">
</TD></TR>
<TR><TD class="clRemLang">
<input type="checkbox" name="RemLang" value="Yes" checked
onChange="javascript:chbRemLang(this.value);">
To remember the chosen language of viewing
</TD>
<TD align="right">
<A HREF="javascript:myLink('rus');">
onMouseOver="javascript:layerWrite('prompt',null,'<table border=\`0\`
cellspacing=\`0\` cellpadding=\`0\`><tr><td width=\`400\`
class=\`clPrompt\` align=\`right\`>Нью-Йорк ОТЛИЧНО<br>Осмотр
достопримечательностей и<br>
Живая музыка<br>Круизы<td></tr></table>');
window.status='Russian'; return true"
onMouseOut="window.status=''; return true"></A>&nbsp;   
```

```

<A HREF="javascript:myLink('eng');"
onMouseOver="javascript:layerWrite('prompt',null,'<table border=\`0\`
cellspacing=\`0\` cellpadding=\`0\`><tr><td width=\`400\`
class=\`clPrompt\` align=\`right\`>New York\'s BEST<br>Sightseeing
and<br>Live Music<br>Cruises<td></tr></table>');
window.status='English'; return true"
onMouseOut="window.status=''; return true"></A>&nbsp;

<A HREF="javascript:myLink('ger');"
onMouseOver="javascript:layerWrite('prompt',null,'<table border=\`0\`
cellspacing=\`0\` cellpadding=\`0\`><tr><td width=\`400\`
class=\`clPrompt\` align=\`right\`>...<td></tr></table>');
window.status='Germanic'; return true"
onMouseOut="window.status=''; return true"></A>&nbsp;

<A HREF="javascript:myLink('spa');"
onMouseOver="javascript:layerWrite('prompt',null,'<table border=\`0\`
cellspacing=\`0\` cellpadding=\`0\`><tr><td width=\`400\`
class=\`clPrompt\` align=\`right\`>...<td></tr></table>');
window.status='Spanish'; return true"
onMouseOut="window.status=''; return true"></A>

</TD></TR></TABLE>
</FORM>
<MAP name="MyMap">
  <AREA alt="Sightseeing Cruises" coords="45,205,188,217"
    href="SightseeingCruises.cfm">
  <AREA alt="Harbor Lights Evening" coords="45,236,188,248"
    href="HarborLightsEvening.cfm">
  <AREA alt="Special Events" coords="45,267,188,280"
    href="SpecialEvents.cfm">
</map>
</CFIF>
</BODY>
</HTML>

```

Результат выполнения данного примера приведен на рис. 7.1. В коде страницы `my_line.cfm` присутствуют ссылки на следующие файлы:

- ❑ `fotoNY.gif` — вид статуи Свободы;
- ❑ `pic1.gif` — предметный сюжет к Кольцевой линии города Нью-Йорка;

- ☐ rus.gif — национальный флаг Российского государства;
- ☐ eng.gif — национальный флаг США;
- ☐ ger.gif — национальный флаг Германии;
- ☐ spa.gif — национальный флаг Испании;
- ☐ commercial.cfm — страница с дополнительной информацией.

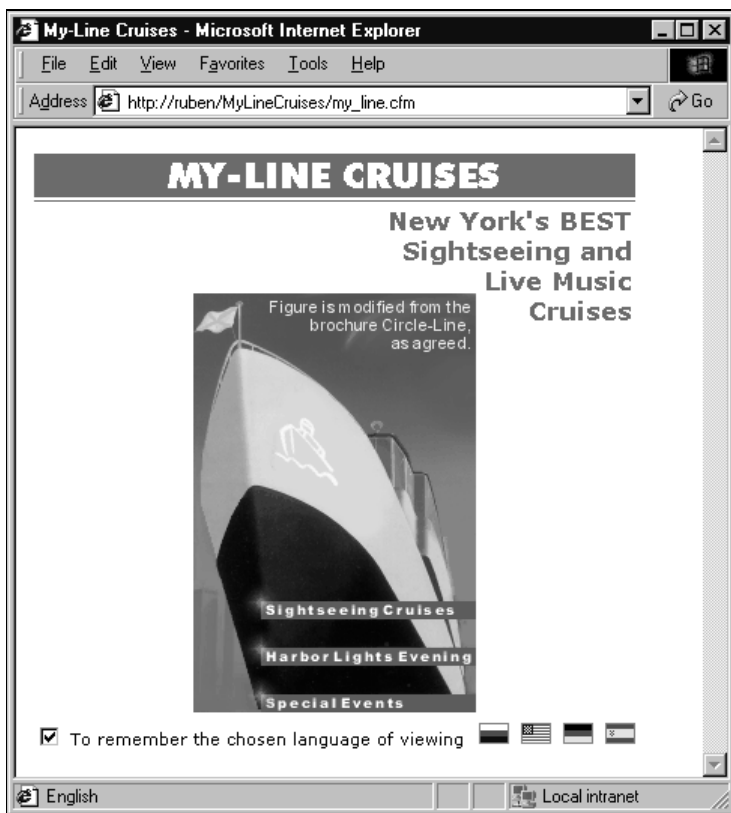


Рис. 7.1. Вид страницы my_line.cfm при первом вхождении

Отметим, что на рис. 7.1, в поле URL-адреса указан Web-сервер ruben, а не localhost или 127.0.0.1, как это освещалось в *главе 6*. Все достаточно просто, обращаться к Web-серверу можно и по его имени. И если ваш сервер называется myweb, то и указывать путь к нему вы можете как **<http://myweb/my-app/>**.

Для нашего примера, выбрав на странице один из национальных флагов, предположим флаг Соединенных Штатов, загрузится англоязычная часть страницы my_line.cfm с указанными параметрами (рис. 7.2). Если флажок **To remember the chosen language of viewing** (Сохранить выбранный язык для

просмотра) будет отмечен, то при следующей загрузке мы сразу окажемся на англоязычной части страницы.

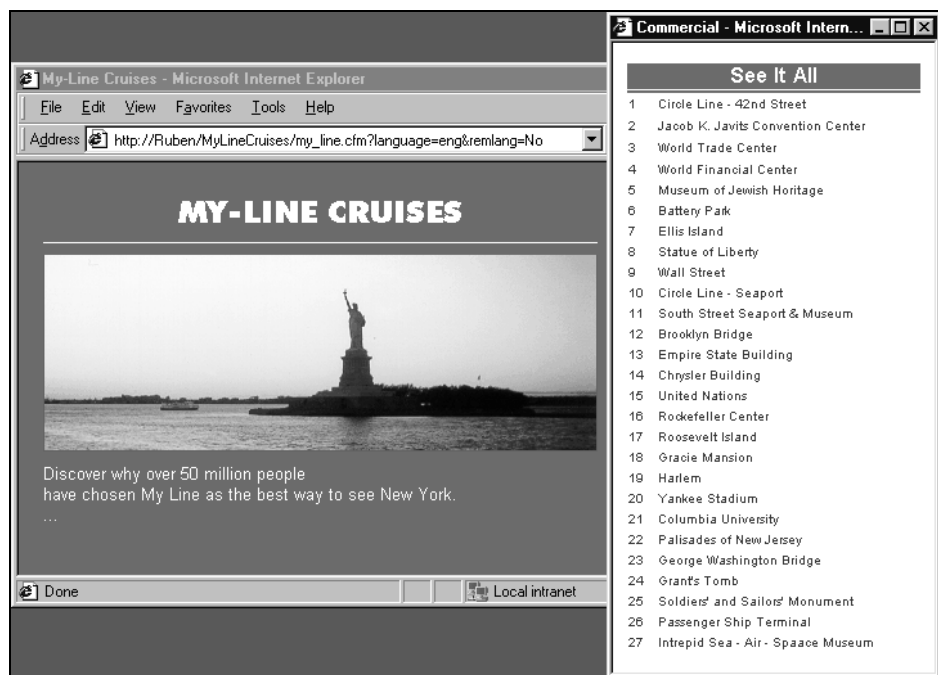


Рис. 7.2. Вид страницы `my_line.cfm` с параметрами

Мы не описываем страницу `commercial.cfm` с дополнительной информацией (на рис. 7.2 она отображена в отдельном окне) в силу простоты кода. Вы можете разработать эту страницу самостоятельно.

Описание кода

Теперь давайте разберем код страницы `my_line.cfm` без учета классических HTML-тегов и вспомогательных текстов.

Таблица 7.2. Анализ кода страницы `my_line.cfm`

Код	Описание
<code><script></code>	Блок функций.
<code>function myLink(lang) {</code>	Функция <code>myLink</code> открывает
<code> linecommand="window.location.href =</code>	страницу <code>my_line.cfm</code> с параметрами
<code> 'my_line.cfm?language='+</code>	<code>language</code> и <code>remlang</code> .

Таблица 7.2 (продолжение)

Код	Описание
<pre>+lang+"&remlang="+ document.form1.RemLang.value+" "; eval(linecommand); } function chbRemLang(value) { if (value=='Yes') { document.form1.RemLang.value='No'; } else { document.form1.RemLang.value='Yes'; } } </script></pre>	Функция <code>chbRemLang</code> используется для переопределения значения элемента <code>RemLang</code> формы <code>form1</code>. Элемент <code>RemLang</code> имеет тип <code>checkbox</code> (флажок), значение которого не зависит от состояния флажка. Это связано с тем, что форму обычно используют с параметром <code>action</code>, где значение флажка передается только в том случае, когда флажок отмечен. Нас это не устраивает, т. к. значение этого элемента мы отправляем в качестве параметра вручную. Вот и приходится "играть" на событии <code>onChange</code>. Если элемент <code>RemLang</code> принимал значение <code>Yes</code>, следовательно, в момент изменения флажка значение данного элемента должно принять противоположное значение. И наоборот
<pre><CFIF isDefined("Cookie.language") OR isDefined("language")></pre>	Если существует <code>Cookie.language</code> или страница включает параметр <code>language</code>, тогда в HTML-страницу помещается код, расположенный ниже вплоть до раздела <code><CFELSE></code>
<pre><CFIF isDefined("Cookie.language")> <CFSET language=#Cookie.language#> </CFIF></pre>	Если существует <code>Cookie.language</code>, определяем переменную <code>language</code> независимо от ее существования
<pre><CFIF isDefined("remlang")> <CFIF remlang eq "Yes"> <CFCOOKIE NAME="language" VALUE="#language#" EXPIRES="7"> </CFIF> </CFIF></pre>	Если существует параметр <code>remlang</code>, равный <code>Yes</code> (что указывает на согласие пользователя с необходимостью запомнить выбранный язык просмотра), устанавливаем <code>cookie</code> равным <code>language</code> на семь дней

Таблица 7.2 (продолжение)

Код	Описание
<pre> <CFSWITCH EXPRESSION=#language#> <CFCASE VALUE="rus"> Русский текст ... </CFCASE> <CFCASE VALUE="eng"> Английский текст ... </CFCASE> <CFCASE VALUE="ger"> Немецкий текст ... </CFCASE> <CFCASE VALUE="spa"> Испанский текст ... </CFCASE> <CFDEFAULTCASE> ... </CFDEFAULTCASE> </CFSWITCH> </pre>	Используем тег <code><CFSWITCH></code> условного оператора. Исходя из значения переменной <code>language</code>, помещаем необходимый текст
<pre> <CFIF isDefined("Cookie.commercial") is False> <CFCOOKIE NAME="commercial" VALUE="1" EXPIRES="12/31/2001"> </pre>	Если не существует <code>Cookie.commercial</code>, устанавливаем cookie равным <code>commercial</code> до 31 декабря 2001 года
<pre> <CFELSE> <CFIF #Cookie.commercial# eq 1> <script> Variable = window.open("commercial.cfm", "windowname", "top=10, left=530, width=250, height=490"); </script> </CFIF> </CFIF> </pre>	Иначе. <code>Cookie.commercial</code> определен и равен 1. Открываем страницу <code>commercial.cfm</code> с дополнительной информацией в отдельном окне

Таблица 7.2 (окончание)

Код	Описание
<CFELSE>	Иначе. <code>Cookie.language</code> не определен или отсутствует параметр <code>language</code> , тогда в HTML-страницу помещается код, расположенный ниже вплоть до раздела окончания тега оператора <code><CFIF></code>
<pre> <FORM method='post' name='form1'> <input type="checkbox" name="RemLang" value="Yes" checked onChange = "javascript:chbRemLang(this.value);"> To remember the chosen language of viewing &nbsp; &nbsp; &nbsp; </FORM> </pre>	Формируем текст первой страницы с формой выбора национального языка для дальнейшей работы
</CFIF>	Конец тега <code><CFIF></code> условного оператора

В качестве самостоятельной работы можете попробовать совместить полученные знания о cookies с примером из предыдущего урока, где при каждом открытии страницы `welcome_flex.cfm` мы принуждали пользователя заполнять данные о себе, что, в общем-то, не всегда разумно, если эти данные можно сохранить у клиента через браузер сразу же при первом запросе. А при очередном открытии страницы считывать однажды введенные значе-

ния cookies. При этом еще раз отметим, что пользователь имеет возможность удалять cookies, размещенные у него на компьютере.

Впрочем, здесь вы можете при определении языка просмотра направлять пользователя на другую заранее сформированную страницу, например на `index_en.cfm` в данном случае с текстом на английском языке.

Существуют и другие способы задания значений cookie, через теги `<META>` языка HTML, JavaScript- и CGI-сценарии.

Резюме

Подведем итог. В данной главе мы использовали CFML-теги: `<CFCOOKIE>`, `<CFSWITCH>`/`<CFCASE>`/`<CFDEFAULTCASE>`.



Операторы CFML

Теперь, когда вы имеете представление о построении страниц с помощью CFML, можно перейти к детальному рассмотрению операторов и выражений языка ColdFusion.

Операторы — это конструкции, выполняющие некоторые действия над одним или несколькими аргументами и выдающие результат. В качестве таких конструкций выступают специальные символы, которые сообщают серверу, что вы хотите выполнить операцию с некоторыми операндами. Типы операций задаются с помощью операторов, а операнды — это переменные, выражения или литералы. Операторы, требующие наличия одного операнда, называются *унарными*. Одни операторы ставятся перед операндом и называются *префиксными*, другие — после операнда, их называют *постфиксными* операторами. Операторы, которые указываются между операндами, называются *инфиксными бинарными операторами*.

В ColdFusion имеется 22 оператора, которые делятся на четыре различных типа: арифметические, операторы отношения или сравнения, логические операторы и строковый оператор.

Арифметические операторы

Арифметические операторы используются для вычислений. Допустимые операнды должны иметь числовые типы. Список операторов приведен в табл. 8.1.

Таблица 8.1. Арифметические операторы

Оператор	Описание
+, -, *, /	Сложение, вычитание, умножение и деление. В случае деления второй операнд не может быть равен нулю
+, -	Унарные арифметические операторы для установки знака числа, положительный (+) или отрицательный (-)

Таблица 8.1 (окончание)

Оператор	Описание
MOD	Оператор вычисления остатка от деления. Результат имеет тот же знак, что и первый операнд. Второй операнд не может быть равен нулю. Например, 10 MOD 3, результат равен 1
\	Целочисленное деление. Второй операнд не может быть равен нулю. Например, 10 \ 3, результат равен 3
^	Возведение в степень. Первый операнд не может быть равен нулю. Например, 10 ^ 3, результат равен 1000

Операторы отношения или сравнения

Для того чтобы можно было сравнивать два значения, в ColdFusion имеется набор операторов, описывающих отношения и равенство. Операторы отношения или сравнения возвращают логическое значение `True` (Истинно) или `False` (Ложно). Список операторов приведен в табл. 8.2.

Таблица 8.2. Операторы отношения или сравнения

Оператор	Описание
IS	Возвращает <code>True</code> , если значения операндов идентичны
IS NOT	Противоположно IS
CONTAINS	Проверяет, содержится ли в первом операнде значение второго операнда, и возвращает <code>True</code> при положительном результате
DOES NOT CONTAIN	Противоположно CONTAINS
GREATER THAN	Проверяет, является ли значение первого операнда больше, чем значение второго операнда, и возвращает <code>True</code> при положительном результате
LESS THAN	Противоположно GREATER THAN
GREATER THAN OR EQUAL TO	Проверяет, является ли значение первого операнда большим или равным значению второго операнда, и возвращает <code>True</code> при положительном результате
LESS THAN OR EQUAL TO	Проверяет, является ли значение первого операнда меньшим или равным значению второго операнда, и возвращает <code>True</code> при положительном результате

Некоторые операторы отношения или сравнения имеют сокращения, которые можно использовать для написания более компактного CFML-кода.

Использование сокращений также избавляет от запоминания длинных названий операторов и сокращает процесс разработки Web-приложения. Список сокращений приведен в табл. 8.3.

Таблица 8.3. Сокращения операторов отношения или сравнения

Оператор	Альтернативные названия
IS	EQUAL, EQ
IS NOT	NOT EQUAL, NEQ
CONTAINS	Отсутствует
DOES NOT CONTAIN	Отсутствует
GREATER THAN	GT
LESS THAN	LT
GREATER THAN OR EQUAL TO	GTE, GE
LESS THAN OR EQUAL TO	LTE, LE

К примеру, в конструкции

```
<CFSET valTest = 10>
<CFIF valTest GREATER THAN OR EQUAL TO 9>
    It is more or equally
<CFELSE>
    It is less
</CFIF>
```

где — GREATER THAN OR EQUAL TO, можно заменить сокращениями GTE или GE:

```
<CFIF valTest GTE 9>
```

или

```
<CFIF valTest GE 9>
```

Результат при этом не изменится.

Булевы операторы

Булевы, или логические операторы, список которых приведен ниже, оперируют только с операндами типа `boolean`. Все бинарные логические операторы воспринимают в качестве операндов два значения типа `boolean` и возвращают результат того же типа. Список операторов приведен в табл. 8.4.

Таблица 8.4. Булевы операторы

Оператор	Описание
NOT	НЕ. Инвертирует логическое значение. Например, NOT True = False и наоборот
AND	И. Возвращает True, если оба операнда имеют значение True, иначе возвращает False. Например, (True AND True) = True, (True AND False) = False
OR	ИЛИ. Возвращает True, если любой из операндов имеет значение True, иначе возвращает False. Например, (True OR False) = True, (False OR False) = False
XOR	Исключающее ИЛИ. Возвращает True, если операнды имеют различные значения, иначе возвращает False. Например, (True XOR True) = False, (False XOR True) = True
EQV	Эквивалентность (противоположность XOR). Возвращает True, если операнды эквивалентны, иначе возвращает False. Например, (True EQV True) = True, (False EQV True) = False
IMP	Импликация. Возвращает False, только когда первый операнд имеет значение True, а второй — False. Иначе возвращает True

Строковый оператор

В эту категорию входит лишь один оператор (табл. 8.5).

Таблица 8.5. Строковый оператор

Оператор	Описание
&	Соединяет символьные выражения. Например, "Анна " & "Андреевна" возвращает — "Анна Андреевна"

Приоритеты операторов

В ColdFusion действует определенный порядок, или приоритет операций. Из элементарной алгебры известно, что операции умножения и деления имеют более высокий приоритет, чем операции сложения и вычитания. В программировании также существует приоритет операций. Ниже приведен список операций в порядке убывания приоритета.

- ☐ унарные +, -
- ☐ ^
- ☐ *, /

- ☐ \
- ☐ MOD
- ☐ +, -
- ☐ &
- ☐ EQ, NEQ, LT, LTE, GT, GTE, CONTAINS, DOES NOT CONTAIN
- ☐ NOT
- ☐ AND
- ☐ OR
- ☐ XOR
- ☐ EQV
- ☐ IMP

Для явной установки приоритета можно использовать круглые скобки (). Вы всегда можете добавить в выражение несколько пар скобок, если у вас есть сомнения по поводу порядка вычислений или вам хочется сделать свой код более читабельным.

Например, пусть необходимо рассчитать осевой момент инерции относительно центральной оси Y для сечения из двух равных прямоугольников. Для определения порядка вычислений скобки расставим следующим образом:

$$J_y = (b^3 * (h - h1)) / 12$$

Здесь b — ширина сечения, h — внешняя высота сечения; $h1$ — внутренняя высота сечения.

В следующей главе рассмотрим переменные и типы, используемые в ColdFusion.

Глава 9



Переменные и типы CFML

В предыдущих главах мы уже сталкивались с применением переменных, а в данной главе рассмотрим более подробно возможности использования переменных в ColdFusion. Но, для начала, познакомимся с понятием *типов данных* и допустимыми типами данных в ColdFusion.

Типы данных

Тип — это классификация, основанная на содержимом и индивидуальности. Тип определяет свойства выражения или переменной таким образом, что вы можете предсказать ее поведение.

В ColdFusion существуют следующие основные типы данных:

- ❑ целочисленный — `Integer`;
- ❑ с плавающей точкой — `Real`;
- ❑ символьный — `String`;
- ❑ логический — `Boolean`;
- ❑ дата — `Date`;
- ❑ время — `Time`.

Кроме того, в ColdFusion доступны несколько более сложные типы:

- ❑ списки — `List`;
- ❑ структуры — `Structure`;
- ❑ массивы — `Array`;
- ❑ запросы — `Query`;
- ❑ СОМ-объекты (Component Object Model) — `COM object`.

Описание типов данных приведено в табл. 9.1.

Таблица 9.1. Описание типов данных

Типы	Описание
Integer	Целое число. Может принимать значения от -2 147 483 648 до 2 147 483 647
Real	Числовые данные с плавающей точкой. Тип предназначен для хранения чисел. Всякий раз, выполняя математические операции, вы имеете дело с той или иной разновидностью числовых значений Диапазон чисел ColdFusion — приблизительно от -1^{300} до 1^{300}. Максимальное число цифр после запятой — 12. Например, 3,14159265359. ColdFusion поддерживает представление чисел в экспоненциальной форме, например, 5E10
String	Символы. Текст, заключенный в одинарные или двойные кавычки. Максимальная длина ограничена количеством доступной памяти в ColdFusion Server
Boolean	Логический тип — результат логического действия. Возможные значения: True (Истина) или False (Ложь). Числовой эквивалент True — 1. Числовой эквивалент False — 0. Символьный эквивалент True — "Yes". Символьный эквивалент False — "No"
Date	Дата и время. Идентифицируют дату и время, где минимальное значение года может быть 100, а максимальное — 9999. Если никакая часть времени не определена, время устанавливается в 12:00am. Вы можете также непосредственно задавать дату в любом из следующих форматов: • "June 01, 1970" • "Jun 01, 1970" • "Jun. 01, 1970" • "06/01/70" • "1970-01-06"
Time	Вы можете непосредственно задавать дату и время в любом из следующих форматов: • "June 01, 1970 02:34:12" • "June 01, 1970 2:34a" • "June 01, 1970 2:34am" • "June 01, 1970 02:34am" • "June 01, 1970 2am" Точность — до секунды

Таблица 9.1 (продолжение)

Типы	Описание
List	Списки — специальный символьный тип, составленный из элементов, отделенных разделителями. Допускается самостоятельное определение разделителя, который может состоять как из одного, так и из нескольких символов. По умолчанию в качестве разделителя принимается запятая (,). Разделители перед первым элементом и после последнего элемента игнорируются. Структура списков плоская, т. е. списки не могут быть вложены друг в друга. Также, списки не могут содержать никакие "пустые" элементы. Однако список может быть пуст, в этом случае он эквивалентен пустому значению — ""
Structure	Вы можете использовать структуры, чтобы создавать и поддерживать пары ключевых значений, обращаться к связанным символьным значениям как к общности, а не как к индивидуальным значениям, или формировать ассоциативные массивы
Array	Массивы — это группы переменных одного типа, доступ к которым осуществляется с помощью общего имени. Хотя функция <code>ArrayNew</code> только поддерживает создание массивов, до третьей размерности, массив не имеет предела на свой размер¹ или максимальное значение. Элементы записываются в массив следующим образом: <pre><CFSET MyNewArray = ArrayNew(1)> <CFSET MyNewArray[1] = "Anjuta"></pre>
Query	В ColdFusion запросы могут быть объявлены как объекты и назначены в качестве переменной запроса: <pre><CFQUERY NAME=MyNewQuery DATASOURCE=Cruise SELECT * FROM TICKET </CFQUERY> <CFSET MySecondQuery = MyNewQuery></pre> В данном случае (в отличие от того же самого действия с массивами) запрос не скопирован. Вместо этого обе переменные указывают на набор записей данных так, что если вы производили изменения в таблице, упомянутой в запросе, первоначальный запрос и запрос <code>MySecondQuery</code> отразят эти изменения. Запросы и переменные не могут одновременно иметь одинаковые имена в пределах одной Web-страницы

¹ Размер и размерность массива — разные понятия. Размерность можно определить как количество направлений в пространстве (например, вектор имеет размерность 1, а таблица (или матрица) — размерность 2). Размер массива — это количество его элементов. — *Ред.*

Таблица 9.1 (окончание)

Типы	Описание
COM object	COM-объекты — невидимые компоненты, инкапсулирующие определенные функциональные возможности. Примерами COM-объектов могут служить элементы ActiveX, OCX, CORBA и ADO-объекты. COM-объекты обычно содержат методы, подобно функциям, которые можно использовать, чтобы выполнить некоторые действия: <pre><CFSET objExcelChart = objCharts.Add()></pre> COM-объекты также обычно характеризуются свойствами. Вы можете считывать и записывать свойства объектов, используя ColdFusion-переменные: <pre><CFSET objExcelChart.Type = -4100></pre>

Переменные

Переменная — это один из элементов хранения информации. Переменные в CFML для хранения данных размещаются в контейнерах, к которым можно обратиться по имени. Это позволяет разрабатывать программы, содержащие переменные, с достаточно наглядным кодом, т. к. вместо неструктурированных переменных программа использует имена, которые разъясняют назначение данных.

Переменные в ColdFusion имеют определенные области действия и назначение и по этим признакам относятся к определенной категории. Эти категории, отражающие префикс, который может быть добавлен к имени переменной, представлены в табл. 9.2.

Таблица 9.2. Описание областей действия переменных

Префикс	Описание
Local	Локальные переменные, создаются с помощью тегов <code><CFSET></code> или <code><CFPARAM></code>
URL	Переменные, отражающие параметры, добавленные к URL-адресу
CGI	Переменные, отражающие CGI-переменные окружения
Cookie	Переменные, отражающие cookies, передаваемые браузеру
Form	Переменные, отражающие значения полей формы, переданные в текущую страницу
Request	Переменные, отражающие значения после обработки запросов на странице

Таблица 9.2 (окончание)

Префикс	Описание
Caller	Переменные источников вызова; предназначены для переменных из шаблонов при указании пользовательских тегов ColdFusion
Attributes	Переменные атрибутов пользовательских тегов
Client	Клиентские переменные
Session	Переменные сессий
Application	Переменные прикладных задач
Server	Серверные переменные

Имена переменных должны начинаться с буквы, которая может сопровождаться любым цифровым символом или символом подчеркивания (`_`). Например, `MyVariable_1` и `MyVariable_2`, в то время как строка `1stMyVariable!` не может быть использована в качестве имени переменной.

Иногда имена переменных могут начинаться с *приставки* (префикса), которая непосредственно является отражением типа переменной. Приставка отделяется от имени переменной точкой (`.`), например, `Form.FirstName`. В этом случае приставка `Form` показывает, что переменная `FirstName` является полем формы, переданной в текущую Web-страницу.

К обязательным префиксам относятся `Attributes`, `Request`, `Session`, `Application`, `Server` и `Caller` (в случае использования пользовательских тегов).

При отсутствии использования приставки в других переменных, ColdFusion совершает поиск значений в следующем порядке:

1. Local

2. CGI

3. File
4. URL

5. Form
6. Cookie

7. Client

Определение переменных

В ColdFusion для определения переменных существуют такие теги, как `<CFSET>` и `<CFPARAM>`. Здесь мы приведем несколько примеров определения переменных с использованием упомянутых тегов. Но обо всем по порядку.

Тег <CFSET>

Одним из вариантов определения переменной в ColdFusion является указание тега `<CFSET>`. Если переменная уже существует, она определяется повторно.

Синтаксис:

```
<CFSET имя_переменной = выражение>
```

Например, для определения переменной `FirstName` со значением "Anjuta" необходимо включить в текущую Web-страницу следующий код:

```
<CFSET FirstName="Anjuta">
```

В следующем примере создаем новый массив `MyCruisesLine` с перечислением круизов:

```
<CFSET MyCruisesLine = ArrayNew(1)>
<CFSET MyCruisesLine[1] = "1-hr Seaport Liberty">
<CFSET MyCruisesLine[2] = "2-hr Semi Circle">
<CFSET MyCruisesLine[3] = "3-hr Full Island">
<CFSET MyCruisesLine[4] = "2-hr Harbor Lights Evening">
<CFSET MyCruisesLine[5] = "Special Events">
```

И в заключение данного раздела представим пример для работы с COM-объектом. Здесь мы создаем COM-объект на основе приложения MS Excel с помощью тега `<CFOBJECT>`, а используя тег `<CFSET>`, определяем свойства и методы этого объекта:

```
<CFOBJECT ACTION="CREATE"
           CLASS="Excel.Application"
           NAME="objExcel"
           TYPE="COM">
<CFSET objExcel.Visible = "False">
<CFSET objWorkBook = objExcel.Workbooks>
<CFSET objOpenedBook = objWorkBook.Add()>
<CFSET objWorkSheets = objExcel.WorkSheets>
<CFSET objWorkSheet = objWorkSheets.Add()>
<CFSET objRange = objExcel.Range("A1:A1")>
<CFSET objRange.value = "MyLineCruises">
<CFSET objWorkSheet.SaveAs("D:\Cruises.xls",Val(1))>
<CFSET objWorkBook.Close()>
<CFSET objExcel.Quit()>
<CFSET objExcel = "Nothing">
```

В результате данного примера из диапазона ячеек A1:A1, которому присваивается имя "MyLineCruises", создается файл MS Excel и сохраняется в каталоге D:\ с именем Cruises.xls.

Тег <CFPARAM>

Еще одним вариантом определения переменной в ColdFusion является использование тега <CFPARAM>, который также обеспечивает возможность указания типа данных и значения по умолчанию, в том числе может применяться для проверки существования переменной.

Синтаксис:

```
<CFPARAM NAME = "variable_name"  
          TYPE = "data_type"  
          DEFAULT = "default_value">
```

Имеются два способа использования тега <CFPARAM>, в зависимости от задачи:

- ☐ применение тега <CFPARAM> только с указанием атрибута имени (NAME), проверяет существование переменной; если переменная не существует, ColdFusion Server прекращает обрабатывать страницу;
- ☐ использование тега <CFPARAM> с атрибутами имени (NAME) и значения по умолчанию (DEFAULT) для проверки существования необязательной переменной; если переменная уже существует, обработка кода продолжается, и значение переменной не изменяется; если переменная не существует, создается переменная с именем, указанным атрибутом NAME и со значением, указанным атрибутом DEFAULT.

Следующий пример показывает, как использовать тег <CFPARAM> для определения переменной, если она отсутствует:

```
<CFPARAM NAME="Form.Gender" DEFAULT="male">
```

Отметим также возможные варианты необязательного атрибута TYPE:

- ☐ any — любое значение;
- ☐ array — любой массив;
- ☐ binary — бинарное значение;
- ☐ boolean — логическое значение;
- ☐ date — значение даты и времени;
- ☐ numeric — числовое значение;
- ☐ query — объект запроса;
- ☐ string — символьное значение;
- ☐ struct — структура;
- ☐ UUID — универсальный уникальный идентификатор (Universally Unique Identifier) в формате "XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXXXXXX", где "X" — шестнадцатеричное число, принимающее значение 0—9 или A—F;
- ☐ ИМЯ — имя существующей переменной.

Например, определим переменную Moneyness с типом boolean:

```
<CFPARAM NAME="Moneyness" DEFAULT=True TYPE="boolean">
```

По умолчанию тип данных переменной принимает значение any.

CGI-переменные

CGI (Common Gateway Interface, стандартный интерфейс шлюза) является стандартом интерфейса связи внешней прикладной программы с информационным сервером типа HTTP или Web. В ColdFusion существует набор стандартных CGI-переменных окружения версии 1.1, которые можно использовать для диагностики запросов (request), сервера (server), посылающего запросы, и клиента (client), принимающего эти запросы. Описание переменных приведено в табл. 9.3—9.5.

Таблица 9.3. CGI-переменные запросов

Наименование	Описание
CGI.CONTENT_LENGTH	Длина в байтах тела запроса
CGI.CONTENT_TYPE	Тип тела запроса
CGI.PATH_INFO	Дополнительная информация о пути, которую передал клиент
CGI.PATH_TRANSLATED	Сервер передает преобразованную версию PATH_INFO, которая включает в себя путь, преобразованный из виртуального в физический
CGI.QUERY_STRING	Строка запроса. Принимает значение при использовании метода GET и подобного
CGI.HTTP_REFERER	Ссылка на страницу при выполнении запроса. Принимает значение при использовании метода GET и подобного
CGI.REQUEST_METHOD	Метод, который был использован для запроса. Для HTTP это GET, POST, HEAD и т. д.
CGI.SCRIPT_NAME	Виртуальный путь к шлюзу текущего запроса

Таблица 9.4. CGI-переменные диагностики сервера

Наименование	Описание
CGI.AUTH_TYPE	Эта переменная определена в том случае, когда запрошенный ресурс требует аутентификации пользователя
CGI.GATEWAY_INTERFACE	Версия CGI-спецификации
CGI.REMOTE_ADDR	IP-адрес удаленного хоста, производящего данный запрос

Таблица 9.4 (окончание)

Наименование	Описание
CGI.REMOTE_HOST	Имя удаленного хоста, производящего запрос. Если сервер не имеет такой информации, он должен задать REMOTE_ADDR, а это поле оставить не установленным
CGI.REMOTE_USER	Переменная определена в том случае, когда запрошенный ресурс требует аутентификации пользователя
REMOTE_IDENT	Если HTTP-сервер поддерживает идентификацию пользователя согласно RFC 931, то переменная будет содержать имя пользователя, полученное от сервера
CGI.SERVER_NAME	Имя хоста, на котором запущен сервер, DNS-имя или IP-адрес в том виде, в котором он представлен в URL
CGI.SERVER_PORT	Номер порта, через который был послан запрос. По умолчанию HTTP-порт 80
CGI.SERVER_PROTOCOL	Имя и версия информационного протокола
CGI.SERVER_SOFTWARE	Название и версия информационного сервера, который отвечает на запрос и запускает шлюз

Таблица 9.5. CGI-переменные диагностики клиента

Наименование	Описание
CGI.CERT_ISSUER	Издатель сертификата
CGI.CLIENT_CERT_ENCODED	Полная информация о сертификате клиента в закодированном виде (Base-64 encoded)
CGI.CERT_SUBJECT	Субъект издательского сертификата
CGI.HTTP_ACCEPT	Список MIME-типов, которые клиент может обработать, как задано в HTTP-заголовках
CGI.HTTP_IF_MODIFIED_SINCE	Переменная предоставляет возможность эффективного обновления информации локальных кэшей с минимумом передаваемой информации
CGI.HTTP_USER_AGENT	Информация о браузере, который использует клиент для отправки запроса

В следующем примере с помощью CGI-переменной HTTP_USER_AGENT мы распознаем тип браузера, и если этот браузер является встроенным в мобильный телефон, который поддерживает технологию WAP, тогда формируем соответствующий код, используя язык WML. В противном случае загру-

жаем Web-страницу `idx.cfm` с кодом, "читаемым" обычными компьютерными браузерами:

```
<CFIF CGI.HTTP_USER_AGENT contains "wap">
  <CFCCONTENT TYPE="text/vnd.wap.wml">
    <?xml version="1.0"?>
    <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
    <wml>
      <card id="start" ontimer="idx.wml">
        <timer value="1"/>
        <p align="center">
          <br/><i>One moment ...</i>
        </p>
      </card>
    </wml>
  <CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=idx.cfm">
  </CFIF>
```

К этому примеру мы еще вернемся в *главе 21*.

Клиентские переменные

Клиентские переменные могут использоваться в таких целях, как сохранение параметров, характеризующих шрифт, цвет, стиль и расположение страницы, а также хранение значения счетчика и последнего времени посещения страницы, в конце концов, контроль результатов опросов, тестов, закупок в электронных магазинах и т. д.

В ColdFusion существует три возможных механизма хранения клиентских переменных.

- ❑ Registry (Реестр) — в этом случае переменные хранятся в реестре операционной системы (system registry). При использовании данного механизма хранения жизненный цикл клиентских переменных равен 10 дням. Однако, имея доступ к Web-странице "ColdFusion Administrator", данное значение можно изменить (рис. 9.1). Этот механизм хранения переменных является значением по умолчанию.
- ❑ Cookie (Ячейка) — название говорит само за себя, подобный механизм подразумевает хранение переменных в cookies, где жизненный цикл переменных лимитирован свойствами cookies.
- ❑ Data Source (Источник данных) — в этом случае хранение клиентских переменных осуществляется в пользовательской базе данных через про-

писанный ODBC, OLE DB или собственный источник данных. Жизненный цикл переменных по умолчанию равен 90 дням, впрочем, и в этом случае данное значение также возможно изменять.

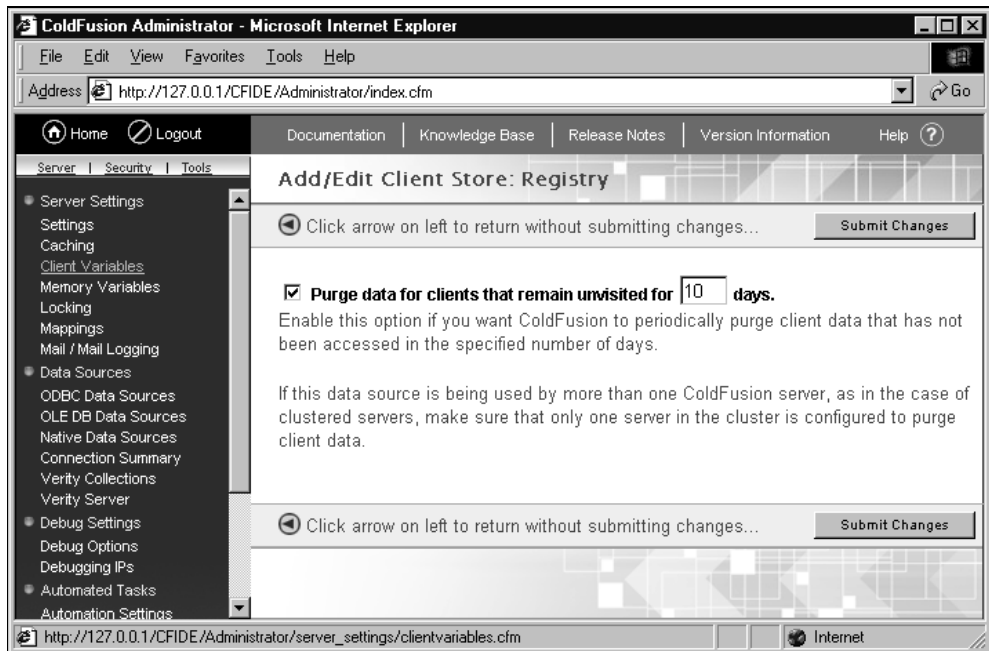


Рис. 9.1. Установка жизненного цикла клиентской переменной для механизма хранения в реестре

За выбор механизма хранения клиентских переменных отвечает атрибут `CLIENTSTORAGE` тега `<CFAPPLICATION>`. Атрибут `CLIENTMANAGEMENT` того же тега позволяет включать или отключать управление клиентскими переменными.

В следующем примере с помощью тега `<CFPARAM>` создадим клиентскую переменную `Client.NameCruise` со значением по умолчанию "1-hr Seaport Liberty":

```
<CFPARAM NAME="Client.NameCruise" DEFAULT="1-hr Seaport Liberty">
```

Если учесть, что в нашем случае механизм хранения переменных не изменялся и принимает значение по умолчанию — Registry (Реестр), то и переменную следует искать в системном реестре. На рис. 9.2 показано окно **Registry Editor** (Редактор реестра) после выполнения примера создания клиентской переменной `Client.NameCruise`.

Как вы заметили, жизненный цикл клиентских переменных достаточно длительный, так что при необходимости принудительного удаления клиентской

переменной можно воспользоваться функцией `DeleteClientVariable` следующим образом:

```
<CFSET IsDeleteSuccessful=DeleteClientVariable("NameCruise")>
IsDeleteSuccessful = <CFOUTPUT>#IsDeleteSuccessful#</CFOUTPUT>
```

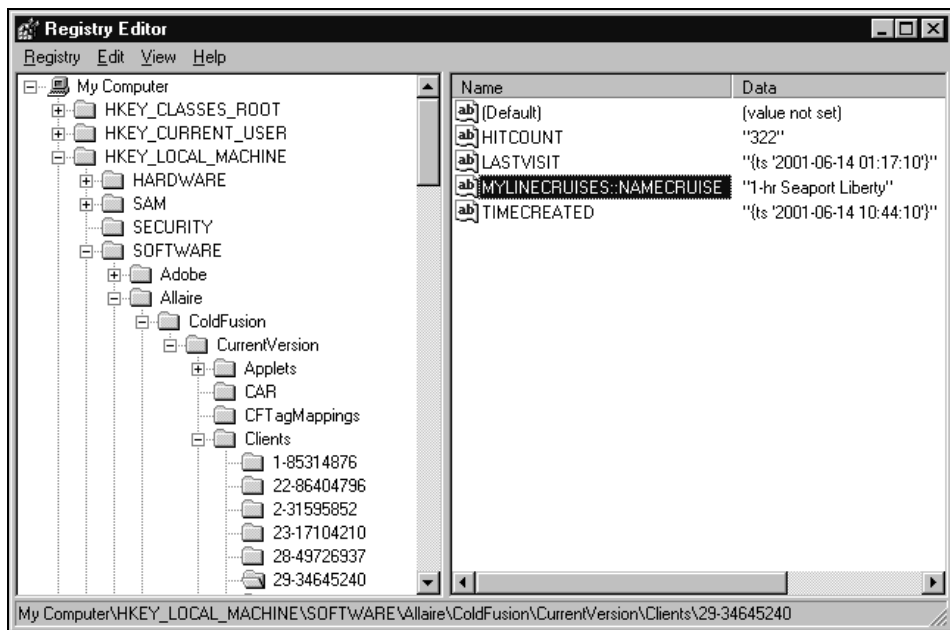


Рис. 9.2. Окно редактора реестра **Registry Editor**

В представленном примере первая строка кода позволяет удалить переменную `NameCruise`, а вторая строка отражает результат переменной `IsDeleteSuccessful`, где значение `"Yes"` указывает на удачное удаление переменной.

Можно, к примеру, воспользовавшись функцией `GetClientVariablesList`, отразить список клиентских переменных:

```
<CFOUTPUT>#GetClientVariablesList()#</CFOUTPUT>
```

В ColdFusion существуют встроенные клиентские переменные, приведенные в табл. 9.6.

Таблица 9.6. Встроенные клиентские переменные

Наименование	Описание
<code>Client.CFID</code>	Идентификатор для каждого клиента, который соединяется с сервером

Таблица 9.6 (окончание)

Наименование	Описание
<code>Client.CFTOKEN</code>	Произвольно сгенерированный номер, являющийся уникальным идентификатором конкретного клиента
<code>Client.URLToken</code>	Комбинация значений <code>CFID</code> и <code>CFTOKEN</code> , разделенных символом <code>&</code> с указанием наименования переменных
<code>Client.LastVisit</code>	Дата и время последнего посещения, сделанного клиентом
<code>Client.HitCount</code>	Количество запросов, выполненных к странице клиентом, идентифицированным под значениями переменных <code>CFID</code> и <code>CFTOKEN</code>
<code>Client.TimeCreated</code>	Записывает дату и время, когда переменные <code>CFID</code> и <code>CFTOKEN</code> были созданы для данного клиента

Работа со встроенными клиентскими переменными возможна при включении атрибута `CLIENTMANAGEMENT` тега `<CFAPPLICATION>`. Для этого достаточно в конфигурационном файле приложения `Application.cfm` записать следующий код:

```
<CFAPPLICATION NAME="MyLineCruises"
                CLIENTMANAGMENT="Yes">
```

Более подробно использование механизма хранения переменных с помощью базы данных и конфигурационного файла `Application.cfm` рассмотрим в *главе 10*.

Переменные сессий

Переменные сессии служат для одновременного фиксирования посещения Web-страницы. Сессия определяется индивидуальным пользователем, в результате чего каждый пользователь имеет отдельную сессию и конкретный набор переменных сессии. Сессия начинается при первом вхождении на Web-участок и формально заканчивается при завершении работы браузера. Таким образом, один пользователь может работать сразу с несколькими сессиями, предварительно запустив браузер необходимое количество раз. Мы говорим о формальном завершении сессии, т. к. сессия имеет период времени, по окончании которого при бездействии пользователя сессия завершается, а, следовательно, и жизненный цикл переменных прекращается. То есть зарегистрировались вы на одном из Web-участков, некоторое время поработали и, не закрывая ваш любимый браузер, отлучились от компьютера, например, ушли на обед. А через определенный промежуток времени сессия завершается. Приходите с обеда, пытаетесь совершить какие-либо действия в окне открытого браузера, но ничего не получается. Все дело в том, что пе-

ременные сессии, часть из которых могли отвечать за доступ к Web-участку, прекратили свое существование.

Переменные сессий могут использоваться в таких целях, как хранение параметров регистрации пользователя, хранение вычислений, основанных на информации из клиентских переменных, хранение временных результатов запросов к данным, в общем, контроль над всеми переменными, требующими сохранения только в течение единой сессии.

В ColdFusion жизненный цикл переменных сессий по умолчанию равен 20 минутам. Однако так же, как и с клиентскими переменными, имея доступ к Web-странице "ColdFusion Administrator", данное значение можно изменить (рис. 9.3).

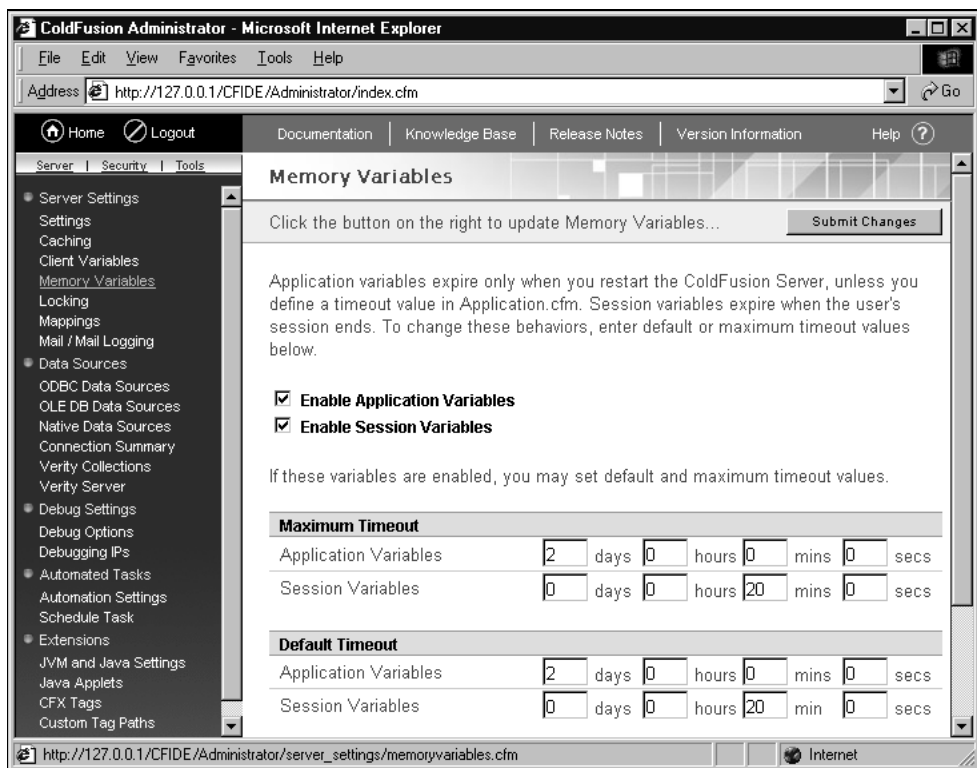


Рис. 9.3. Страница "ColdFusion Administrator", раздел **Memory Variables**

Более того, данный период можно указать программным методом, все в том же пока мало известном конфигурационном файле приложения Application.cfm, используя атрибут SESSIONTIMEOUT все того же тега <CFAPPLICATION>.

Скажем, для указания периода жизни переменных сессий, равным 25 минутам, и включения управления этими переменными следует в конфигурационном файле приложения `Application.cfm` записать следующий код:

```
<CFAPPLICATION NAME="MyLineCruises"
    SESSIONTIMEOUT="#createTimeSpan(0,0,25,0)#"
    SESSIONMANAGEMENT="Yes">
```

Для примера объявим переменную сессии `Login` следующим образом:

```
<CFSET Session.Login = "true">
```

Надо отметить, что принудительное удаление той или иной переменной сессии возможно при вызове функций `StructDelete` или `StructClear`. Таким образом, осуществить удаление переменной `Login` можно следующим образом:

```
<CFLOCK TIMEOUT="30" SCOPE="Session" TYPE="Exclusive">
    <CFSET StructDelete(Session, "Login")>
    <CFIF IsDefined("Session.Login")>
        Variable still exists.
    <CFELSE>
        Variable is successfully removed!
    </CFIF>
</CFLOCK>
```

В приведенном примере мы использовали блокировку посредством тега `<CFLOCK>`, что является правилом хорошего тона при работе в сети. Более подробно применение блокировок рассмотрим в *главе 10*.

В ColdFusion существуют встроенные переменные сессий, перечень которых приведен в табл. 9.7.

Таблица 9.7. Встроенные переменные сессий

Наименование	Описание
<code>Session.CFID</code>	Идентификатор для каждого клиента, который соединяется с сервером
<code>Session.CFTOKEN</code>	Произвольно сгенерированный номер, являющийся уникальным идентификатором конкретного клиента
<code>Session.URLToken</code>	Комбинация значений переменных <code>CFID</code> и <code>CFTOKEN</code> , разделенных символом <code>&</code> с указанием наименования переменных
<code>Session.SessionID</code>	Идентификатор сессии, представляющий комбинацию значений наименования приложения <code>CFID</code> и <code>CFTOKEN</code> , разделенных символом подчеркивания (<code>_</code>)

Работа со встроенными переменными сессий возможна при включении атрибута `SESSIONMANAGEMENT` тега `<CFAPPLICATION>`.

Переменные сессий хранятся в оперативной памяти сервера.

Переменные прикладных задач

Переменные прикладных задач (далее будем называть их прикладными переменными) связаны с единым приложением и доступны всем клиентам данного приложения. Прикладные переменные могут использоваться в таких целях, как хранение учета обращений к текущему приложению, хранение результатов запросов к данным, к которым могут обращаться все клиенты приложения, хранение наименований источников данных, а также хранение заголовка приложения и стандартных сообщений и т. д.

Для использования прикладных переменных необходимо создать конфигурационный файл с именем `Application.cfm` и добавить туда определение имени приложения, используя атрибут `NAME` тега `<CFAPPLICATION>`. Указанные определения в конфигурационном файле приложения распространяются на все прилегающие к этому файлу страницы приложения. То есть, загружая ту или иную страницу ColdFusion, система ищет файл `Application.cfm` вплоть до корневого каталога приложения и при первой удачной попытке содержимое найденного файла помещается в начало загруженной ColdFusion-страницы.

В ColdFusion жизненный цикл прикладных переменных по умолчанию равен 2-м дням. Однако, имея доступ к Web-странице "ColdFusion Administrator", данное значение можно изменить. И так же, как и с переменными сессий, период жизненного цикла можно указать программным методом, в конфигурационном файле приложения `Application.cfm`, используя атрибут `APPLICATIONTIMEOUT` тега `<CFAPPLICATION>`.

Например, для указания периода жизни прикладных переменных равным 4-м часам следует в конфигурационном файле приложения `Application.cfm` записать следующий код:

```
<CFAPPLICATION NAME="MyLineCruises"
APPLICATIONTIMEOUT="#createTimeSpan(0,4,0,0)#"
SESSIONMANAGEMENT="Yes">
```

Для примера объявляем прикладную переменную `textFieldSize` следующим образом:

```
<CFPARAM NAME="Application.textFieldSize" DEFAULT="25">
```

Следующий пример позволяет вывести список прикладных переменных с использованием тега `<CFLOOP>`, определив эти переменные в качестве коллекции:

```
<CFLOCK TIMEOUT="30" SCOPE="Application" TYPE="ReadOnly">
<CFLOOP COLLECTION=#Application# ITEM=itemID>
<CFOUTPUT>ID=#itemID# Value=#Application[itemID]#</CFOUTPUT><br>
```



```
</CFLOOP>  
</CFLOCK>
```

В этом примере мы использовали блокировку прикладных переменных только на чтение.

Удаление переменных прикладных задач производится по аналогии с удалением переменных сессий. К примеру, для удаления ранее объявленной переменной `textFieldSize` можно использовать следующую конструкцию:

```
<CFLOCK TIMEOUT="30" SCOPE="Application" TYPE="Exclusive">  
    <CFSET StructDelete(Application, "textFieldSize")>  
</CFLOCK>
```

В качестве встроенных прикладных переменных в ColdFusion существует переменная `Application.ApplicationName`, принимающая значение атрибута `NAME` тега `<CFAPPLICATION>`, объявленного в конфигурационном файле `Application.cfm`.

Серверные переменные

Серверные переменные не привязаны к какому-либо приложению (application) или клиенту, а относятся непосредственно к ColdFusion Server, и по этой причине к ним можно обращаться всем клиентам или из разных приложений данного сервера. Эти переменные хранятся в оперативной памяти сервера до тех пор, пока тот не будет приостановлен.

Например, объявим серверную переменную `CountUserActive` с использованием тега `<CFSET>`:

```
<CFSET Server.CountUserActive = 0>
```

Удаление и просмотр серверных переменных осуществляется по аналогии с переменными сессий и переменными прикладных задач. Скажем, удаление серверной переменной `CountUserActive` можно осуществить следующим способом:

```
<CFLOCK TIMEOUT="30" SCOPE="Server" TYPE="Exclusive">  
    <CFSET StructDelete(Server, "CountUserActive")>  
</CFLOCK>
```

В ColdFusion существуют встроенные серверные переменные, приведенные в табл. 9.8.

Таблица 9.8. Встроенные серверные переменные

Наименование	Описание
<code>Server.ColdFusion.ProductName</code>	Название продукта ColdFusion Server
<code>Server.ColdFusion.ProductVersion</code>	Номер версии запущенного сервера, типа 5.0.0

Таблица 9.8 (окончание)

Наименование	Описание
Server.ColdFusion.ProductLevel	Уровень продукта, например Enterprise
Server.ColdFusion.SerialNumber	Последовательный номер, назначенный при установке
Server.ColdFusion.Locales	Язык и регион поддерживаемый сервером, например, English (US)
Server.OS.Name	Название операционной системы, например Windows NT
Server.OS.AdditionalInformation	Дополнительная информация, обеспеченная операционной системой. Например, номер Service Pack
Server.OS.Version	Версия операционной системы, например 4.0
Server.OS.BuildNumber	Номер сборки, определенный операционной системой, типа 1381

Используйте серверные переменные, когда хотите организовать доступ к данной переменной из любого другого клиентского соединения вашего сервера. Не храните слишком большое количество серверных переменных.

В табл. 9.9 приведем сравнение переменных по различным характеристикам.

Таблица 9.9. Сравнение переменных

Тип переменной	Возможность блокировки	Способ хранения	Период жизненного цикла, по умолчанию
Client	Нет	Registry (Реестр)	10 дней
		Cookie (Ячейка)	90 дней
		Data Source (Источник данных)	
Session	Да	ОЗУ сервера	20 минут
Application	Да	ОЗУ сервера	2 дня
Server	Да	ОЗУ сервера	До завершения работы сервера

Просмотр переменных

В процессе разработки приложений достаточно часто необходимо просматривать значения объявленных переменных, и если для формирования CFML-страниц вы не используете редактор ColdFusion Studio, позволяющий отлаживать страницы посредством встроенного механизма, то для этих це-

лей можно использовать программный метод просмотра существующих переменных.

Для отладки переменных можно применять тег `<CFDUMP>`, позволяющий наглядно демонстрировать значения простых переменных, запросов, массивов, структур и WDDX-переменных, созданных с помощью тега `<CFWDDX>`.

К примеру, вспомним код страницы `my_line.cfm`, детально описанной в главе 7 (см. листинг 7.1). Поместим в него следующую строку:

```
<CFDUMP var="#Cookie#">
```

Этим мы определим необходимость проверки всех существующих на странице ячеек (`Cookie`). На рис. 9.4 показан результат обращения к странице `my_line.cfm` с применением тега `<CFDUMP>`.



Рис. 9.4. Результат диагностики переменных `Cookie`

Следующий пример позволяет создать список HTTP-заголовков для просмотра их значений, используя функцию `GetHttpRequestData()`.

Листинг 9.1. Код страницы `GetHttpRD.cfm`

```
<HTML>
```

```
<HEAD>
```

```

<TITLE>GetHttpRequestData</TITLE>
<LINK rel="STYLESHEET" type="text/css" href="my_style.css">
</HEAD>
<BODY>
<CFSET HttpRData = GetHttpRequestData()>
<CFOUTPUT>
  <TABLE bgcolor="E1E1E1" cellpadding = "0" cellspacing = "0">
    <TR>
      <TD bgcolor="666699" align="center" class="clTitle">
        Name HTTP-Request</TD>
      <TD bgcolor="666699" align="center" class="clTitle">
        Value</TD>
    </TR>
    <CFLOOP COLLECTION = #HttpRData.headers# ITEM = "http_name">
      <TR>
        <TD class="clText">#http_name#</td>
        <TD class="clText">#StructFind (HttpRData.headers,
http_name)#</td>
      </TR>
    </CFLOOP>
  </TABLE>
</CFOUTPUT>
</BODY>
</HTML>

```

На рис. 9.5 представлен результат отображения страницы GetHttpRequestData.cfm в браузере.

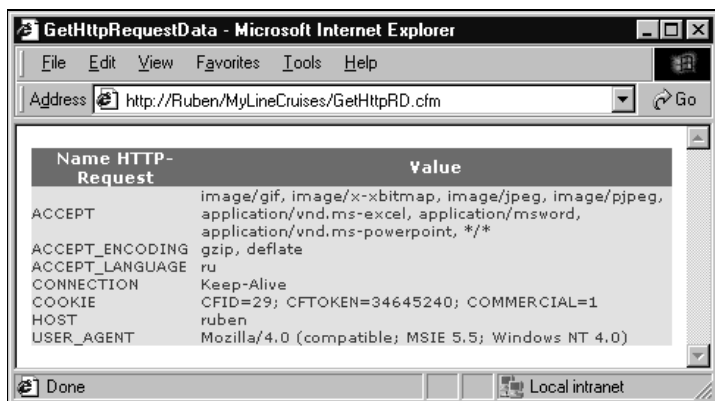


Рис. 9.5. Результат использования функции GetHttpRequestData()

Отметим лишь то, что в данном примере для функции `GetHttpRequestData()` мы использовали вхождение `headers`, что, собственно говоря, и позволило нам отфильтровать переменные HTTP-заголовков. Также эта функция предоставляет возможность использования вхождений `content`, `method` и `protocol`.

Проверка переменных

Одним из главных правил при работе с переменными является то, что прежде чем совершить какие-либо действия по отношению к той или иной переменной, необходимо произвести проверку наличия таковой. Мы уже не раз приводили примеры с использованием функции `IsDefined`, позволяющей придерживаться данного правила. И все же еще раз обратим ваше внимание на пример, обеспечивающий проведение проверки в данном случае переменной `Session.Login`:

```
<CFIF IsDefined("Session.Login")>
    Variable is present.
<CFELSE>
    Variable is absent.
</CFIF>
```

Всегда ли мы уверены, что над переменной можно произвести ряд действий, требующих представления этой переменной в формате даты. Конечно же, нет. К примеру, вы предоставляете пользователю форму для ввода данных о себе с последующим занесением результата заполнения в базу данных, где одним из полей является день рождения. И далеко не всегда пользователь в состоянии с первого раза правильно заполнить данное поле, причем по разным причинам. Выходов из этого положения может быть несколько: контроль над полем на уровне JavaScript, размещение на форме ввода специальных инструментов типа "spinner", контролирующих ввод данных. Однако мы будем использовать возможности ColdFusion.

В следующем примере объявим переменную `Form.BirtDayMyMam` и тем самым подменим процесс ввода этой переменной. И уже на второй строке кода с помощью функции `IsDate` проверим, можно ли преобразовать эту переменную, используя одну из функций по работе с датами (в нашем случае функция форматирования даты — `DateFormat`):

```
<CFSET Form.BirthDayMyMam = "4-19-1940">
<CFIF IsDate(Form.BirthDayMyMam)>
    <CFOUTPUT>
        #DateFormat(Form.BirthDayMyMam, "mmm-dd-yyyy")#
    </CFOUTPUT>
<CFELSE>
    Return on the form of input
</CFIF>
```

Результатом данного примера является следующая строка:

Apr-19-1940

Вот еще один похожий пример, только уже позволяющий проверить в данном случае переменную `Form.CostAdults` на наличие числового формата. Здесь мы также подменяем процесс ввода на объявление переменной с использованием тега `<CFSET>`:

```
<CFSET Form.CostAdults = "17.9">
<CFIF IsNumeric(Form.CostAdults)>
    <CFOUTPUT>
        #NumberFormat(Form.CostAdults, '$____.____')#
    </CFOUTPUT>
<CFELSE>
    Return on the form of input
</CFIF>
```

В приведенном выше примере используется функция `NumberFormat` с необязательным параметром маски (`mask`), позволяющая форматировать числовые значения. И результатом этого примера является строка:

\$ 17.90

Очередной пример позволяет нам определить, можно ли полученную переменную привести к логическому формату данных. И здесь мы воспользовались тегом `<CFSET>` для подмены процесса ввода переменной `Form.MyYesNo`:

```
<CFSET Form.MyYesNo = 1>
<CFIF IsBoolean(Form.MyYesNo)>
    <CFOUTPUT>
        #YesNoFormat(Form.MyYesNo)#
    </CFOUTPUT>
<CFELSE>
    Return on the form of input
</CFIF>
```

В приведенном выше примере используется функция `YesNoFormat`, позволяющая преобразовывать логическое значение или число к виду `Yes` или `No`. Результатом этого примера является строка:

Yes

Следует отметить, что параметр функции `IsBoolean` может быть как в виде числового, так и в виде символического значения в отличие от предыдущих функций `IsDate` и `IsNumeric`, где в качестве параметра должны присутствовать символические значения. При значении параметра, равным нулю, `False`

или No функция `IsBoolean` возвращает `False`, в противном случае возвращается `True`.

В заключение представим полный код страницы `signup.cfm`, предназначенной для регистрации пользователя с применением вышеописанных функций, позволяющих контролировать ввод данных на уровне ColdFusion.

Листинг 9.2. Код страницы `signup.cfm`

```
<HTML>
<HEAD>
  <META name="Author" content="Ruben Akhayan">
  <TITLE>Sing up</TITLE>
  <style type="text/css">
    <!--
    .txt {
      padding : 0px;
      font-weight : normal;
      font-family : Verdana, Geneva, Arial;
      font-size : 11px;
      color : White;
    }
    .txtError {
      padding : 0px;
      font-weight : normal;
      font-family : Verdana, Geneva, Arial;
      font-size : 11px;
      color : #FFC0CB;
    }
    INPUT {
      padding : 0px;
      font-weight : normal;
      font-family : Verdana, Geneva, Arial;
      font-size : 11px;
      color : Blue;
    }
    .note {
      color : #F5FFFA;
    }
  -->
</style>
```

```
</HEAD>
<BODY bgcolor="<cfoutput>#application.BGColorPage#</cfoutput>">
  <CFIF isDefined("Form.login_use")>
    <CFSET Disuccessful = 1>
    <CFSET msgError = "Warning Error!">

    <CFIF Len(Form.login_use) le 5>
      <CFSET msgError = msgError &
        "<br>Login should consist from more than 5 symbols">
      <CFSET Disuccessful = 0>
    </CFIF>
    <CFIF Len(Form.pass_use) le 4>
      <CFSET msgError = msgError &
        "<br>Password should consist from more than 4 symbols">
      <CFSET Disuccessful = 0>
    </CFIF>
    <CFIF IsNumeric(Form.pass_use)>
      <CFSET msgError = msgError &
        "<br>Password should not consist only of numbers">
      <CFSET Disuccessful = 0>
    </CFIF>
    <CFIF Form.pass_use not equal Form.confirm>
      <CFSET msgError = msgError &
        "<br>Password and Confirm, should be identical">
      <CFSET Disuccessful = 0>
    </CFIF>
    <CFIF Len(Form.fname_use) le 0>
      <CFSET msgError = msgError &
        "<br>First Name, should not be empty">
      <CFSET Disuccessful = 0>
    </CFIF>
    <CFIF Len(Form.lname_use) le 0>
      <CFSET msgError = msgError &
        "<br>Last Name, should not be empty">
      <CFSET Disuccessful = 0>
    </CFIF>
    <CFIF not IsDate(Form.bday_use)>
      <CFSET msgError = msgError &
        "<br>Birthday, should have a format dd-mm-yyyy">
```



```

    <CFSET Disuccessful = 0>
</CFIF>
<CFIF YesNoFormat(Disuccessful) equal "No">
    <CFSET login = #Form.login_use#>
    <CFSET fname = #Form.fname_use#>
    <CFSET lname = #Form.lname_use#>
    <CFSET bday = #Form.bday_use#>
    <CFSET email = #Form.email_use#>
<CFELSE>
    <CFSET msgError="">
</CFIF>
<CFELSE>
    <CFSET login="">
    <CFSET fname="">
    <CFSET lname="">
    <CFSET bday="">
    <CFSET email="">
    <CFSET msgError="">
    <CFSET Disuccessful = 0>
</CFIF>
<CFIF YesNoFormat(Disuccessful) equal "Yes">
    <CFOUTPUT>
        <p class="txt">
            Data input successful<br>
            Hi dear #Form.fname_use# #Form.lname_use#
        </p>
    </CFOUTPUT>
<CFELSE>
    <FONT color="White"><h2>Sign up</h2></font>
    <FORM action='signup.cfm' METHOD='POST' name='form1'>
    <TABLE border='0' cellspacing='5' cellpadding='0'>
        <TR><TD colspan="2" align="center" class="txtError">
            <CFOUTPUT>#msgError#</CFOUTPUT></TD></TR>
        <TR><TD width='100' align="right" class="txt">
            Login</td><td class="note">
                <input name='login_use' size='17'
                    value="<CFOUTPUT>#login#</CFOUTPUT>">&nbsp;*</td></TR>
        <TR><TD align="right" class="txt">
            Password</td><td class="note">

```

```

        <input type='password' name='pass_use'
            size='17'>&nbsp;*</TD></TR>
<TR><TD align="right" class="txt">
    Confirm</td><td class="note">
        <input type='password' name='confirm'
            size='17'>&nbsp;*</TD></TR>
<TR><TD align="right" class="txt">
    First Name</td><td class="note">
        <input name='fname_use' size='17'
            value="<CFOUTPUT>#fname#</CFOUTPUT>">&nbsp;*</TD></TR>
<TR><TD align="right" class="txt">
    Last Name</td><td class="note">
        <input name='lname_use' size='17'
            value="<CFOUTPUT>#lname#</CFOUTPUT>">&nbsp;*</TD></TR>
<TR><TD align="right" class="txt">
    Birthday</td><td class="note">
        <input name='bday_use' size='10'
            value="<CFOUTPUT>#bday#</CFOUTPUT>">&nbsp;
            dd-mm-yyyy&nbsp;*</TD></TR>
<TR><TD align="right" class="txt">
    E-mail</TD><TD class="note">
        <input name='email_use' size='27'
            value="<CFOUTPUT>#email#</CFOUTPUT>"></TD></TR>
<TR><TD align="right">
        <input type='submit' value='Send'></TD><TD>
        <input type='reset' value='Clear'></TD></TR>
</TABLE>
</FORM>
</CFIF>
</BODY>
</HTML>

```

На рис. 9.6 показан результат выполнения страницы `signup.cfm` с преднамеренно введенными ошибочными значениями.

В приведенном выше примере мы использовали еще одну функцию — `Len`, позволяющую определить количество символов указанной переменной в качестве параметра. Что же касается занесения результатов регистрации в базу данных, то об этом поговорим в одной из следующих глав.

The screenshot shows a web browser window with a 'Sign up' form. The form has a title 'Sign up' and a warning message: 'Warning Error! Login should consist from more than 5 symbols Password should not consist only of numbers'. The form fields are: Login (CFRub), Password (empty), Confirm (empty), First Name (Ruben), Last Name (Akhayan), Birthday (06-06-1970), and E-mail (Ruben@ipr.ru). There are 'Send' and 'Clear' buttons at the bottom. The browser's address bar shows 'signup.cfm'.

Рис. 9.6. Результат выполнения страницы signup.cfm

Резюме

Подведем итог. В данной главе использовались:

- ❑ CFML-теги: `<CFPARAM>`, `<CFOBJECT>`, `<CFCONTENT>`, `<CFAPPLICATION>`, `<CFLOCK>`, `<CFLOOP>`, `<CFDUMP>`;
- ❑ CFML-функции: `DeleteClientVariable`, `GetClientVariablesList`, `StructDelete`, `GetHttpRequestData`, `IsDate`, `DateFormat`, `IsNumeric`, `NumberFormat`, `IsBoolean`, `YesNoFormat`, `Len`.

Глава 10



Управление приложением

Приложение ColdFusion представляет собой набор Web-страниц, работающих совместно. Приложения могут быть простыми, как гостевая книга, или сложными, как полноценная торговая система с каталогами, покупательскими корзинами и отчетами. Вы можете объединять индивидуальные приложения для разработки развитых Web-систем, создавая специальный конфигурационный файл `Application.cfm`, размещая его в корневом каталоге приложения. Все другие конфигурационные файлы в приложении хранятся в каталогах ниже корневого.

Структура приложения ColdFusion основана на четырех основных компонентах:

- ☐ настройки уровня приложения и его функций;
- ☐ управление состоянием клиента;
- ☐ обработка ошибок;
- ☐ интеграция безопасности Web-сервера.

При помощи этих компонентов можно легко объединять ваши страницы прикладных приложений ColdFusion в сложные Web-приложения.

ColdFusion предлагает свойства прикладного уровня (Application-level), которые помогут вам контролировать настройки, переменные параметры и характеристики, доступные во всем приложении. Как только вы определите приложение, можете использовать характеристики прикладного уровня наряду со всеми другими возможностями ColdFusion.

Планирование структуры приложения

Важным шагом в проектировании приложения ColdFusion является планирование структуры его каталогов. Прежде, чем начать создавать приложение, назначьте для него корневой каталог. Страницы приложения могут храниться в подкаталогах.

Загружая ту или иную страницу ColdFusion, система ищет файл `Application.cfm` во всех каталогах, вплоть до корневого каталога приложения,

и при первой удачной попытке содержимое найденного файла помещается в начало загруженной ColdFusion-страницы.

Файл `Application.cfm` выполняется перед каждой прикладной страницей, которой он управляет. Вы также можете указать файл `OnRequestEnd.cfm`, который будет выполняться после каждой прикладной страницы в том же приложении.

ColdFusion ищет файл `OnRequestEnd.cfm` в том же каталоге, что и файл `Application.cfm` текущей страницы приложения. Файл `OnRequestEnd.cfm` никогда не будет выполняться, если он находится в другом каталоге, а также, если на данной странице имеется ошибка или исключение или если данная страница выполняет теги `<CFABORT>` или `<CFEXIT>`.

На рис. 10.1 представлена условная схема, отражающая внедрение файлов `Application.cfm` и `OnRequestEnd.cfm`.

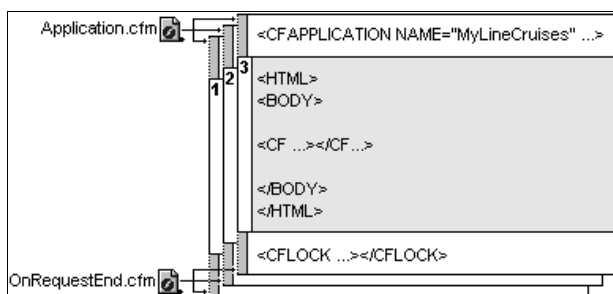


Рис. 10.1. Условная схема внедрения файлов `Application.cfm` и `OnRequestEnd.cfm`

Так же, как файл `Application.cfm` нужно записывать с заглавной буквы `A`, так и имя файла `OnRequestEnd.cfm` следует писать с заглавными буквами `O`, `R`, и `E`.

Определение корневого каталога для страниц приложения имеет ряд преимуществ:

- ❑ *разработка* — приложение легче разрабатывать и поддерживать, т. к. файлы на странице приложения хорошо организованы;
- ❑ *мобильность* — приложение легко переместить на другой сервер или часть сервера без изменения кодов каких-либо файлов на странице приложения;
- ❑ *настройки уровня приложения* — страницы приложения, подпадающие под один корневой каталог, имеют общие уровневые настройки и функции;
- ❑ *безопасность* — страницы приложения, подпадающие под один корневой каталог, имеют общие уровневые защитные настройки Web-сервера.

Вы можете использовать один файл `Application.cfm` для вашего приложения или различные файлы `Application.cfm` для управления отдельными секциями приложения.

На рис. 10.2 представлены два подхода схемы каталогов к применению структуры приложения.

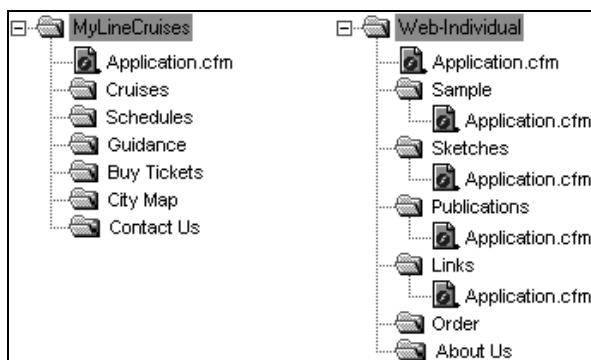


Рис. 10.2. Два подхода схемы каталогов: один файл Application.cfm для всего приложения (слева) и различные файлы Application.cfm для отдельных секций приложения (справа)

В примере слева существует проект под названием MyLineCruises, который использует один файл Application.cfm, расположенный в корневом каталоге приложения и предназначенный для обработки всех запросов на страницах приложения.

Иллюстрация справа показывает, как проект Web-Individual использует настройки в индивидуальных файлах Application.cfm, чтобы задать обработку для приложений ColdFusion на "местном" уровне. Здесь страницы каталогов Order и About Us обрабатываются с помощью настроек файла Application.cfm из корневого каталога, а каждый из каталогов Sample, Sketches, Publications и Links имеет собственный файл Application.cfm.

Пример файла Application.cfm

Прежде чем продемонстрировать пример конфигурационного файла приложения Application.cfm, детально разберем атрибуты тега <CFAPPLICATION> (табл. 10.1), частично затронутого в предыдущей главе.

Таблица 10.1. Описание атрибутов тега <CFAPPLICATION>

Атрибут	Описание	Возможные значения
NAME	Позволяет определить имя приложения. Обязательный атрибут для переменных приложений и сессий и необязательный — для клиентских переменных	До 64 символов

Таблица 10.1 (окончание)

Атрибут	Описание	Возможные значения
APPLICATIONTIMEOUT	Позволяет определить жизненный цикл прикладных переменных. Необязательный атрибут	#CreateTimeSpan(d,h,m,s) # *
CLIENTMANAGEMENT	Дает возможность включать или отключать управление клиентскими переменными. Необязательный атрибут	Yes No (по умолчанию)
CLIENTSTORAGE	Отвечает за выбор механизма хранения клиентских переменных. Необязательный атрибут	Registry (по умолчанию) Cookie Data Source Name
SESSIONMANAGEMENT	Позволяет включать или отключать управление переменными сессии. Необязательный атрибут	Yes No (по умолчанию)
SESSIONTIMEOUT	Позволяет определить жизненный цикл переменных сессии. Необязательный атрибут	#CreateTimeSpan(d,h,m,s) # *
SETCLIENTCOOKIES	Если атрибут имеет значение Yes, ColdFusion автоматически посылает браузеру cookies <CFID> и <CFTOKEN>. Необязательный атрибут	Yes (по умолчанию) No
SETDOMAINCOOKIES	Позволяет устанавливать cookies <CFID> и <CFTOKEN> для домена. Необязательный атрибут. Для приложений, работающих на уровне кластеров, необходимо использовать значение Yes	Yes No (по умолчанию)

* d — дни, h — часы, m — минуты, s — секунды.

В листинге 10.1 приведен пример содержимого файла Application.cfm, где помимо определения основных характеристик приложения объявляются прикладные переменные для последующего их использования на страницах текущего проекта.

Листинг 10.1. Код конфигурационного файла Application.cfm

```
<CFAPPLICATION NAME="MyLineCruises"
APPLICATIONTIMEOUT=""#CreateTimeSpan(0,1,0,0)#"
```

```
SESSIONTIMEOUT="#CreateTimeSpan(0,0,2,0)#"
SESSIONMANAGEMENT="Yes"
CLIENTMANAGEMENT="Yes"
SETCLIENTCOOKIES="Yes"
CLIENTSTORAGE="Registry">

<!-- BGColor -->
<CFPARAM NAME="application.BGColorWin" DEFAULT="666699">
<CFPARAM NAME="application.BGColorPage" DEFAULT="White">
<!-- Database configuration -->
<CFPARAM NAME="application.dsn" DEFAULT="Cruise">
<!-- SMTP -->
<CFPARAM NAME="application.smtp" DEFAULT="127.0.0.1">
<!-- POP -->
<CFPARAM NAME="application.pop" DEFAULT="127.0.0.1">
<!-- Message Error-->
<CFPARAM NAME="application.msgErrorLogin"
    DEFAULT="You have entered not the correct data!">
<CFPARAM NAME="application.msgError1"
    DEFAULT="E-Mail(Login) should consist from more than 5 symbols,
where there should be a symbol @">
<CFPARAM NAME="application.msgError2"
    DEFAULT="Password should consist from more than 4 symbols">
<CFPARAM NAME="application.msgError3"
    DEFAULT="Password should not consist only of numbers">
<CFPARAM NAME="application.msgError4"
    DEFAULT="Password and Confirm, should be identical">
<CFPARAM NAME="application.msgError5"
    DEFAULT="First Name, should not be empty">
<CFPARAM NAME="application.msgError6"
    DEFAULT="Last Name, should not be empty">
<CFPARAM NAME="application.msgError7"
    DEFAULT="Birthday, should have a format dd-mm-yyyy">
```

Приведенный пример нельзя воспринимать как идеальный, хотя бы по той причине, что мы еще ни слова не проронили об обработке ошибок, где одним из методов является использование конфигурационного файла приложения. Но об этом позже (см. главу 15).

Тег <CFINCLUDE>

Для каждой прикладной страницы ColdFusion всегда обрабатывается только один файл Application.cfm. Присутствие файла Application.cfm — это, по существу, наличие скрытого тега <CFINCLUDE>. Если этот файл присутствует в древе каталогов, то не включить его невозможно. Следовательно, это идеальное местоположение, чтобы установить переменные уровня приложения.

Когда требуемая прикладная страница имеет тег <CFINCLUDE>, указывающий на дополнительную прикладную страницу, ColdFusion не производит другой поиск по древу каталогов, основываясь на включенной странице приложения. Это важная для понимания особенность. После открытия требуемой прикладной страницы ColdFusion ищет файл Application.cfm только один раз.

К примеру, создадим файл-шаблон с наименованием MyHEAD.cfm для последующего включения во все необходимые нам страницы следующим образом:

```
<CFINCLUDE TEMPLATE = "../MyHEAD.cfm">
```

Здесь путь указан так, что поиск созданного нами шаблона MyHEAD.cfm будет осуществляться в каталоге на один уровень выше, куда собственно говоря мы его и поместим. Однако если на одном уровне со страницей, куда был помещен тег <CFINCLUDE>, находится файл Application.cfm, то именно этот файл и будет включен в код страницы независимо от существования такого же файла конфигурации уже на одном уровне с шаблоном MyHEAD.cfm.

Механизм хранения клиентских переменных

Как мы уже отмечали в предыдущей главе, в ColdFusion существует три возможных механизма хранения клиентских переменных.

- ❑ Registry (Реестр) — в этом случае переменные хранятся в реестре операционной системы (system registry). Механизм хранения применяется по умолчанию. При его использовании жизненный цикл клиентских переменных равен 10 дням.
- ❑ Cookie (Ячейка) — механизм подразумевает хранение переменных в cookies.
- ❑ Data Source (Источник данных) — в этом случае хранение клиентских переменных осуществляется в пользовательской базе данных через прописанный ODBC, OLE DB или собственный источник данных. Жизненный цикл переменных по умолчанию равен 90 дням, Впрочем, и в этом случае данное значение также возможно изменять.

На рис. 10.3 показана страница администратора "ColdFusion Administrator", открытая в разделе **Server | Server Setting | Client Variables** (Сервер | Пара-

метры сервера | Клиентские переменные), где предоставляется возможность визуально изменять механизм хранения клиентских переменных.

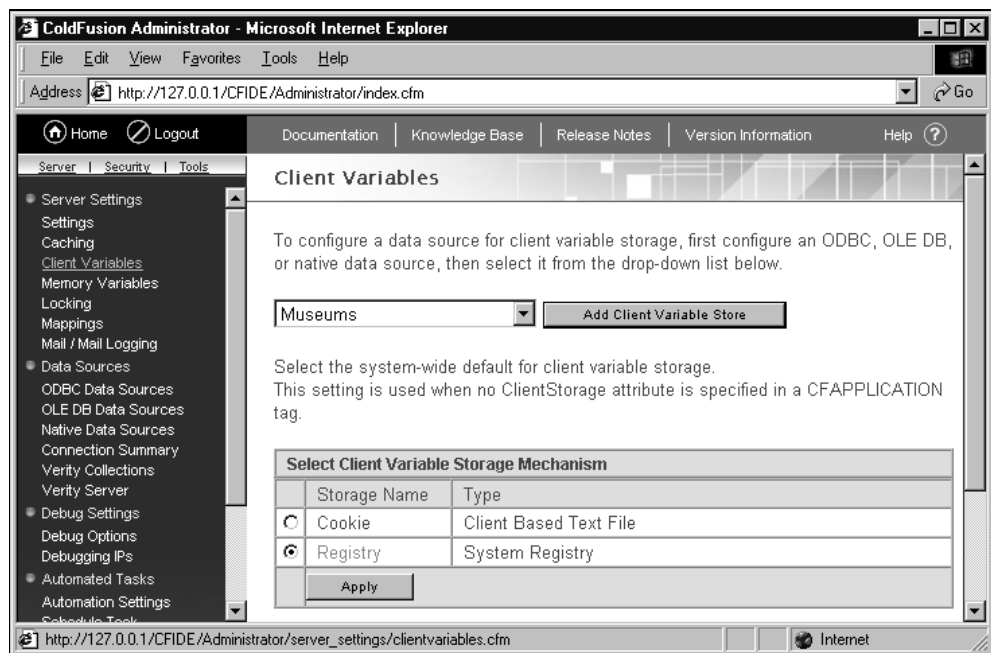


Рис. 10.3. Страница "ColdFusion Administrator", раздел **Client Variables**

Хранение клиентских переменных в системном реестре предполагает расположение объявленных переменных в следующем разделе My Computer \ HKEY_LOCAL_MACHINE \ SOFTWARE \ Allaire \ ColdFusion \ CurrentVersion \ Clients \.

Позволим себе более детально описать механизм хранения клиентских переменных с использованием базы данных, т. к. этот механизм, по мнению автора, требует особого пояснения.

Хранение клиентских переменных в базе данных

Для использования базы данных в качестве хранилища клиентских переменных в первую очередь, следует сформировать эту базу и создать к ней источник данных (Date Source). Причем средства ColdFusion позволяют формировать необходимую структуру данных при непосредственном выборе механизма хранения переменных. Однако организовать подобную структуру данных можно и вручную. Для этого следует создать таблицы CDATA и CGLOBAL.

Описание структуры таблицы `CDATA` приведено в табл. 10.2.

Таблица 10.2. Структура таблицы `CDATA`

Поле	Формат	Описание
<code>cfid</code>	Текстовое поле длиной до 20 символов	Хранит комбинацию двух встроенных переменных <code>CFID</code> (идентификатор клиента) и <code>CFTOKEN</code> (произвольно сгенерированный номер, являющийся уникальным идентификатором конкретного клиента)
<code>app</code>	Текстовое поле длиной до 64 символов	Хранит имя приложения
<code>data</code>	Мето-поле	Хранит перечень объявленных клиентских переменных, отделенных друг от друга символом <code>#</code> . Например, <code>"MYFIRST_CV=Values#"</code>

Описание структуры таблицы `CGLOBAL` приведено в табл. 10.3.

Таблица 10.3. Структура таблицы `CGLOBAL`

Поле	Формат	Описание
<code>cfid</code>	Текстовое поле до 20 символов	Хранит комбинацию двух встроенных переменных <code>CFID</code> и <code>CFTOKEN</code>
<code>data</code>	Мето-поле	Хранит комбинацию трех встроенных переменных: <code>HitCount</code> (количество запросов к странице, произведенных клиентом), <code>LastVisit</code> (дата и время последнего посещения страницы клиентом) и <code>TimeCreated</code> (дата и время, когда <code>CFID</code> и <code>CFTOKEN</code> были созданы для данного клиента)
<code>lvisit</code>	Поле даты/времени	Хранит дату и время последнего посещения страницы клиентом

После создания базы данных следует сформировать источник данных (`Date Source`), как это было описано в главе 1. Для нашей базы данных мы создаем источник с именем `CF_CLIENT`. На рис. 10.4 представлен пример подготовленной базы данных `CF_CLIENT`, в нашем примере на платформе сервера БД — Sybase SQL Anywhere.

Следующим шагом на странице администратора "ColdFusion Administrator" в разделе **Server | Server Setting | Client Variables** (Сервер | Параметры сервера | Клиентские переменные) необходимо выбрать созданный источник и нажать на кнопку **Add Client Variable Store** (Добавить хранилище клиентских переменных). На рис. 10.5 показана страница администратора "ColdFusion Administrator" в режиме диалога для добавления нового хранилища клиентских переменных.

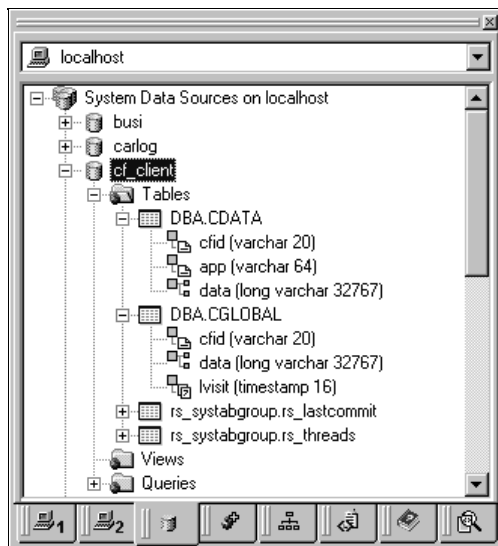


Рис. 10.4. Пример базы данных CF_CLIENT для хранения клиентских переменных

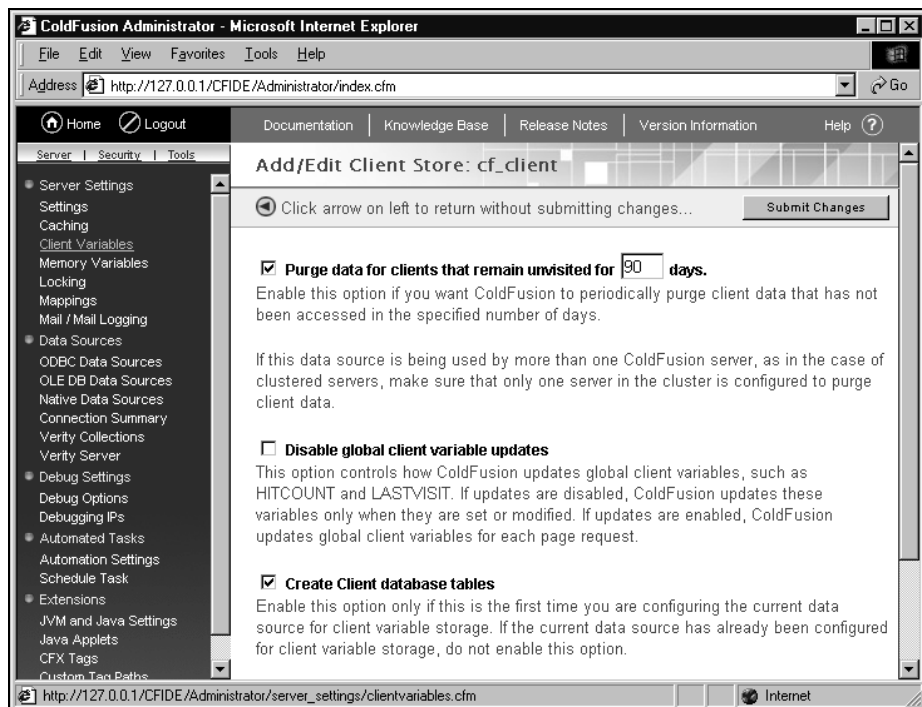


Рис. 10.5. Режим диалога для добавления нового хранилища клиентских переменных

В предоставленном диалоге можно установить очистку данных клиента, которые остаются неиспользуемыми в течение определенного времени, так же в этом диалоге можно включать или отключать модернизацию глобальных клиентских переменных, и как мы уже отмечали, здесь же допустимо включить или отключить автоматическое формирование таблиц с необходимыми полями в указанной базе данных.

Поскольку мы уже заранее сформировали необходимые таблицы для хранения клиентских переменных, флажок, отвечающий за создание этих таблиц (**Create Client database tables** (Создать таблицы базы данных для клиентских переменных)), сбрасываем. После чего нажимаем кнопку **Submit Changes** (Отправить изменения), тем самым подтверждаем добавление нового хранилища клиентских переменных.

На рис. 10.6 показан результат добавления нового хранилища. Здесь же в качестве текущего хранилища клиентских переменных выбрано только что добавленное хранилище.

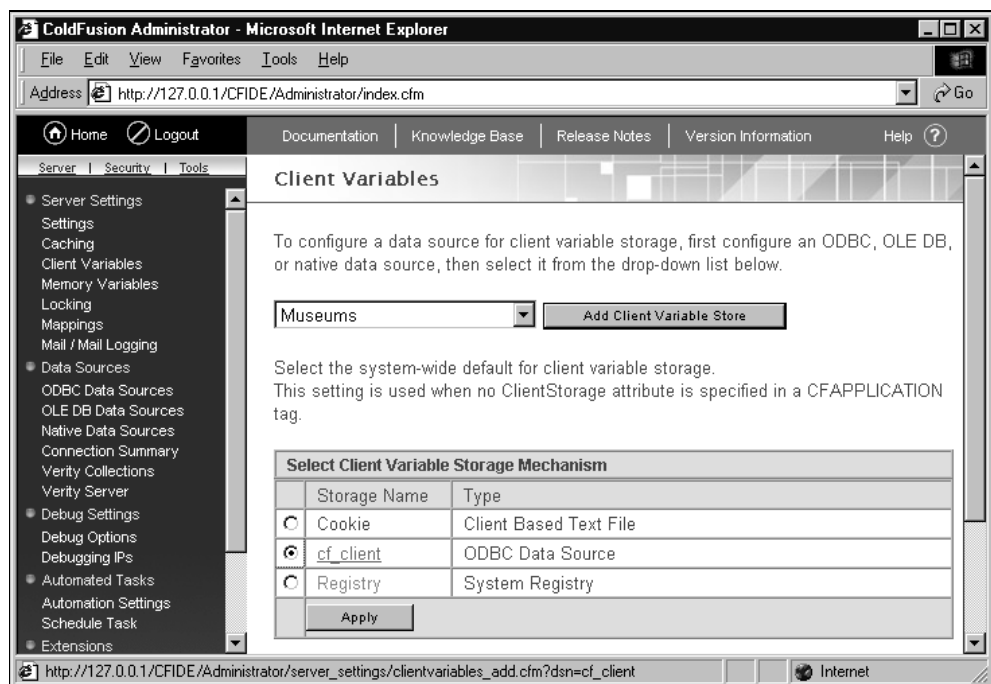


Рис. 10.6. Результат добавления нового хранилища клиентских переменных

Надо отметить, что выбранное хранилище распространяется на приложения, не включающие атрибут `CLIENTSTORAGE` тега `<CFAPPLICATION>`, о котором уже так много было сказано.

Например, для определенного приложения можно индивидуально указать хранилище клиентских переменных, воспользовавшись тегом `<CFAPPLICATION>` следующим образом:

```
<CFAPPLICATION NAME="MyLineCruises"
    APPLICATIONTIMEOUT="#CreateTimeSpan(0,1,0,0)#"
    SESSIONTIMEOUT="#CreateTimeSpan(0,0,2,0)#"
    SESSIONMANAGEMENT="Yes"
    CLIENTMANAGEMENT="Yes"
    SETCLIENTCOOKIES="Yes"
    CLIENTSTORAGE="CF_CLIENT">
```

В этом случае для приложения "My-LineCruises" клиентские переменные будут храниться в созданной нами базе данных `CF_CLIENT`.

Для просмотра данных можно воспользоваться любым приложением, позволяющим строить запросы к базам данных. На рис. 10.7 представлен пример просмотра базы данных `CF_CLIENT` с использованием дизайнера представлений (**View Designer**) из MS Visual FoxPro, позволяющего просматривать содержимое Мемо-полей в отдельных окнах.

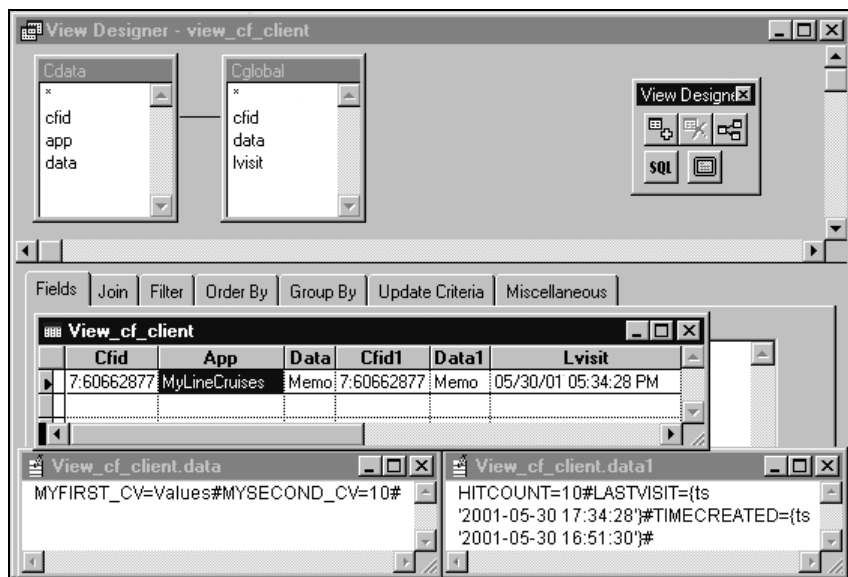


Рис. 10.7. Просмотр базы данных `CF_CLIENT`

Из примера видно, что клиент с идентификатором 7 последний раз посетил страницу приложения 30 мая 2001 года, а всего подобных посещений было

произведено 10. При этом в базе данных хранятся две клиентские переменные: MYFIRST_CV со значением Values и MYSECOND_CV со значением 10.

Использование блокировок <CFLOCK>

В примерах, демонстрирующих работу с переменными сессий, прикладными переменными и серверными переменными, мы уже отмечали использование тега <CFLOCK>, предназначенного для организации блокировок в ColdFusion.

Понимая, что с одним и тем же приложением могут работать сразу несколько пользователей, существует вероятность что, к одним и тем же данным или, например, к одной и той же прикладной переменной может последовать несколько обращений в один промежуток времени, что приведет к нежелательному результату.

Отказ от использования блокировок способен привести к следующим непредсказуемым и нежелательным последствиям:

- ❑ потере данных;
- ❑ возникновению сообщений об ошибках;
- ❑ остановке ColdFusion Server;
- ❑ переполнению памяти;
- ❑ неустойчивой работе операционной системы.

Для блокировки, гарантирующей целостность данных, необходимо использовать тег <CFLOCK>. Блокировать данные можно как для монопольного использования данных — *эксклюзивно (exclusive lock)*, так и для многократного выполнения запросов — *только для чтения (read only lock)*.

Вернемся к одному из наших примеров, где применяется эксклюзивная блокировка для последующего удаления прикладной переменной textFieldSize:

```
<CFLOCK TIMEOUT="30" SCOPE="Application" TYPE="Exclusive">
  <CFSET StructDelete(Application, "textFieldSize")>
</CFLOCK>
```

Здесь для указания типа блокировки существует атрибут TYPE. Необязательный атрибут SCOPE позволяет определить такие области воздействия, как Session, Application или Server. Атрибут SCOPE является взаимоисключающим с атрибутом NAME, который определяет наименование блокировки. Заметим также, что в данном примере присутствует обязательный атрибут TIMEOUT, предназначенный для определения максимального периода времени (в секундах), в течение которого будет совершена попытка блокировки выполнения команд — в данном случае, команды удаления переменной.

Если попытка блокировки оказывается неудачной, дальнейшее поведение будет зависеть от значения необязательного атрибута `THROWONTIMEOUT`. Атрибут `<THROWONTIMEOUT>` определяет условие прерывания и может принимать значения `Yes` или `No`. При значении `Yes`, которое является значением по умолчанию, генерируется исключительная ситуация, чтобы обеспечить уведомление о прерывании. При значении `No` выполнение кода продолжается за рамками тега `</CFLOCK>`.

На странице "ColdFusion Administrator", показанной на рис. 10.8, в списке установок сервера **Server Settings** (Параметры сервера) находится пункт **Locking** (Блокировка), позволяющий управлять установками блокировок.

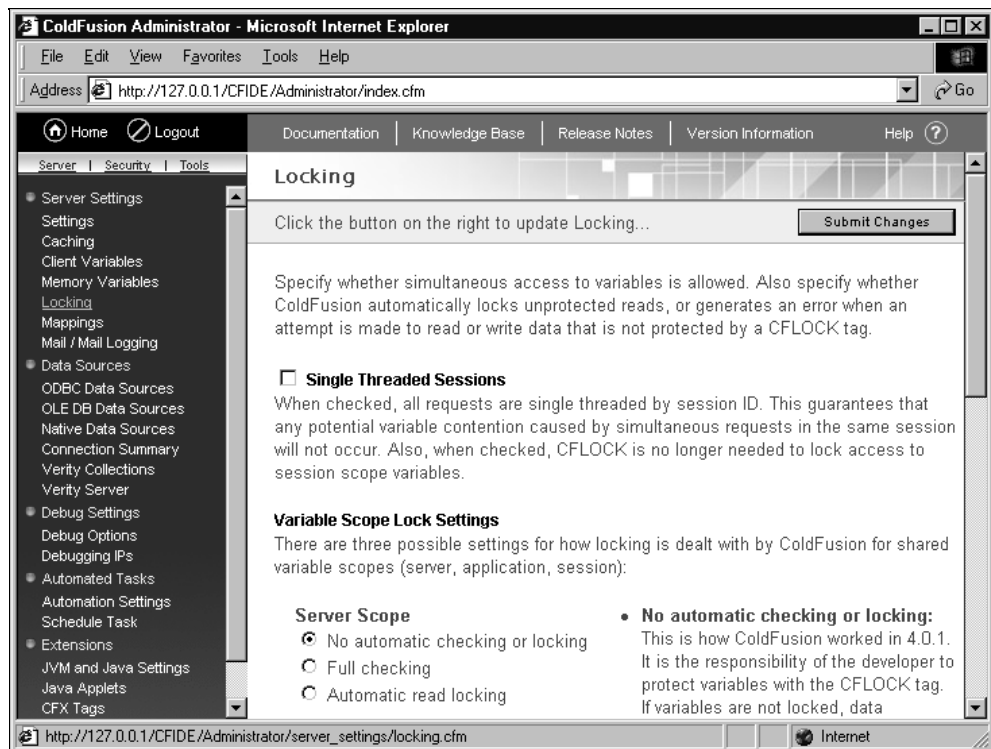


Рис. 10.8. Страница "ColdFusion Administrator", раздел **Locking**

Здесь вам предоставляется возможность указать тип автоматической блокировки для каждого типа переменных: `Server`, `Application` и `Session`.

☐ **No automatic checking or locking** (Не проверять и не устанавливать блокировку автоматически). ColdFusion не контролирует использование блокировок и не предотвращает неверного доступа к разделяемым переменным. Этот вариант наиболее эффективен для завершенных проектов,

когда вы уверены, что все ваши страницы полностью поддерживают все правила блокировки данных.

- ❑ **Full checking** (Полная проверка). ColdFusion генерирует исключительную ситуацию, когда приложение пытается использовать любую переменную в существующих рамках без применения блокировки. Вариант полной проверки наиболее полезен при отладке страниц. Это позволит обнаружить неосторожно открытый доступ к переменным.
- ❑ **Automatic read locking** (Автоматическая блокировка на чтение). Если в приложении производится попытка чтения без использования блокировок, то ColdFusion автоматически блокирует переменную на чтение и, таким образом, не позволяет вносить какие-либо изменения до окончания процесса чтения. Если же в приложении осуществляется попытка записи без использования блокировок и включением данного режима, ColdFusion генерирует исключительную ситуацию. Этот вариант может быть пригодным, если у вас уже существует проект без использования блокировок должным образом. В таком случае необходимо будет блокировать код только там, где производится запись.

Замечание

Не следует использовать включение такого типа автоматической блокировки, как **Full checking** (Полная проверка) в сочетании с применением в блокировке атрибута `NAME`.

Также на странице "ColdFusion Administrator" существует флажок **Single Threading Sessions** (Последовательная обработка сессий), сброшенный по умолчанию. Если установить эту опцию, тогда каждая сессия рассматривается сервером как единственная. В результате запрос от одного клиента должен закончиться прежде, чем следующий клиент сможет его осуществить. В этом случае нет необходимости блокировать переменные сессий, однако страницы, основанные на фреймах, необходимо устранить, чтобы предотвратить одновременное использование запросов из составных фреймов.

Использование `<CFLOCK>` по отношению к CFML-конструкциям, которые способствуют изменениям разделенных данных, гарантирует последовательную модификацию этих данных, так же, как например, использование `<CFLOCK>` при манипуляции с файлами обеспечивает успешную модернизацию этих файлов.



Работа с данными

Необходимо подчеркнуть, что если вам доводилось разрабатывать программы по обработке баз данных, используя язык SQL, то изучение данного материала не составит для вас особой сложности.

Замечание

Все примеры в данной главе основаны на разработанной нами базе данных `CRUISE`, описанной в разд. "Разработка базы данных" гл. 5. В примерах могут встречаться ссылки на основные файлы в рамках проекта "My-Line Cruises", коды которых представлены в *приложении 1*.

Как уже отмечалось ранее, ColdFusion позволяет работать с данными через драйверы ODBC, OLE DB, а также использовать собственные драйверы для Oracle, Sybase, Informix, DB2. Для реализации представленных здесь примеров мы предварительно создали источник ODBC с именем `Cruise` в соответствии с наименованием базы данных.

Напомним, что нами была разработана физическая модель базы данных `CRUISE` для сайта, знакомящего туристов с возможными круизами и экскурсиями, позволяющего бронировать и распространять билеты на текущие круизы.

Также отметим, что при перемещении физической модели `CRUISE` непосредственно на сервер БД Sybase SQL Anywhere было предложено изменить название таблицы `SCHEDULE` в связи с требованиями конкретного сервера БД. Поэтому далее таблица, содержащая расписание, будет именоваться как `SCHEDULE_`. В таком случае окончательный список таблиц, с которыми предстоит работать, выглядит так:

- ❑ круизы (экскурсии) — `TOUR`;
- ❑ промежуточные пункты назначения — `DESTINATION`;
- ❑ маршруты — `ROUTE`;
- ❑ расписание — `SCHEDULE_`;
- ❑ сезоны — `SEASON`;
- ❑ бронированные билеты — `TICKET`;
- ❑ клиенты (путешественники) — `TRAVELER`.

В ColdFusion существует тег <CFQUERY>, позволяющий передавать SQL-запросы в базу данных как для выполнения определенных действий, так и для извлечения результирующих данных. Это не единственный тег в ColdFusion, предназначенный для работы с базами данных. Однако тег <CFQUERY> можно определить как основной и требующий отдельного внимания.

Тег <CFQUERY>

Синтаксис:

```
<CFQUERY NAME = "query_name"
    DATASOURCE = "ds_name"
    DBTYPE = "type"
    DBSERVER = "dbms"
    DBNAME = "database name"
    CONNECTSTRING = "connection string"
    USERNAME = "username"
    PASSWORD = "password"
    MAXROWS = "number"
    BLOCKFACTOR = "blocksize"
    TIMEOUT = "milliseconds"
    CACHEDAFTER = "date"
    CACHEDWITHIN = "timespan"
    PROVIDER = "COMProvider"
    PROVIDERDSN = "datasource"
    DEBUG = "Yes | No">
```

SQL-предложения

```
</CFQUERY>
```

В табл. 11.1 представлено описание атрибутов тега <CFQUERY>.

Таблица 11.1. Описание атрибутов тега <CFQUERY>

Атрибут	Описание
NAME	Имя запроса. Обязательный атрибут. Должно начинаться с буквы, может включать буквы, цифры и символ подчеркивания. Пробелы недопустимы
DATASOURCE	Источник данных, к которому обращен запрос. Обязательный атрибут для всех значений атрибутов DBTYPE кроме DBTYPE="query" и DBTYPE="dynamic"

Таблица 11.1 (продолжение)

Атрибут	Описание
DBTYPE	Тип драйвера базы данных. Необязательный атрибут. Возможные значения: • <code>dynamic</code> — тип соединения с источником данных ODBC, который не определен в "ColdFusion Administrator". При использовании этого значения следует определить всю информацию о связи ODBC с атрибутом <code>CONNECTSTRING</code>; • <code>query</code> — использует существующий запрос как источник данных. В этом случае нет необходимости определять атрибут <code>DATASOURCE</code>, вместо этого следует использовать имя существующего запроса в качестве имени таблицы; • <code>ODBC</code> — ODBC-драйвер (по умолчанию); • <code>Oracle73</code> — позволяет использовать собственный драйвер БД Oracle 7.3. В этом случае на компьютере, где установлен ColdFusion Server, должна быть установлена клиентская часть Oracle 7.3.4.0.0 (или более поздняя версия); • <code>Oracle80</code> — дает возможность использовать собственный драйвер БД Oracle 8.0. Соответственно должна быть установлена клиентская часть Oracle 8.0 (или более поздняя версия); • <code>Sybase11</code> — позволяет использовать собственный драйвер БД Sybase System 11. Соответственно должна быть установлена клиентская часть Sybase System 11.1.1 (или более поздняя версия); рекомендуется с исправлениями ebf 7729; • <code>OLEDB</code> — поставщик OLE DB; • <code>DB2</code> — использование собственного драйвера БД DB2 5.2; • <code>Informix73</code> — использование собственного драйвера БД Informix 7.3
DBSERVER	Имя сервера БД. Необязательный атрибут. Применяется для собственных драйверов и <code>SQLLEDB</code>. При определении <code>DBSERVER</code> игнорируется установка источника данных
DBNAME	Имя БД (только для Sybase System 11 и <code>SQLLEDB</code>). Необязательный атрибут. При определении <code>DBNAME</code> игнорируется БД по умолчанию, установленная в источнике данных
CONNECTSTRING	Строка соединения для отправления к ODBC-серверу. Требуется для <code>DBTYPE = "dynamic"</code>, необязательный атрибут для всех других типов. При соединении с источником данных, определенным в "ColdFusion Administrator", можно использовать этот атрибут, чтобы определить дополнительные детали соединения или переопределить информацию о соединении. Если вы динамически соединяетесь с источником данных, указанным в атрибуте <code>DATASOURCE</code>, определяя <code>DBTYPE="dynamic"</code>, строка соединения должна определить все требуемые параметры соединения ODBC

Таблица 11.1 (окончание)

Атрибут	Описание
USERNAME	Имя пользователя. Необязательный атрибут. При определении данного атрибута значение USERNAME, указанное в источнике данных, игнорируется
PASSWORD	Пароль пользователя. Необязательный атрибут. При определении данного атрибута значение PASSWORD, указанное в источнике данных, игнорируется
MAXROWS	Определяет максимальное количество записей для возвращаемого результата. Необязательный атрибут. Может принимать значение -1, что позволяет не ограничивать возвращаемое количество записей
BLOCKFACTOR	Максимальное количество записей для одновременной выборки (fetch) с сервера. Необязательный атрибут. Диапазон от 1 (по умолчанию) до 100. Этот атрибут применяется для собственных драйверов Oracle и ODBC-драйверов. Некоторые драйверы ODBC могут динамически уменьшать размер данного блока
TIMEOUT	Максимальное время ожидания выполнения запроса в миллисекундах. Необязательный атрибут. TIMEOUT поддерживается SQL Server 6.x и более поздними версиями. Минимально и максимально допустимые значения зависят от драйвера
CACHEDAFTER	Значение даты. Необязательный атрибут. Позволяет использовать кэширование данных запроса, если дата запроса оригинала принимает более позднее значение даты, установленной с помощью CACHEDAFTER. Эффективно только если в "ColdFusion Administrator" разрешено кэширование запросов. Чтобы применять кэширование данных, текущий запрос должен использовать то же самое SQL-предложение, источник данных, имя запроса, имя пользователя, пароль и тип драйвера
CACHEDWITHIN	Промежуток, использующий функцию ColdFusion CreateTimeSpan. Необязательный атрибут. Позволяет применять кэширование данных запроса, если дата запроса оригинала попадает в промежуток времени, который определяется функцией CreateTimeSpan. Эффективно, только если в "ColdFusion Administrator" разрешено кэширование запросов. Чтобы использовать кэширование данных, текущий запрос должен содержать то же самое SQL-предложение, источник данных, имя запроса, имя пользователя, пароль и тип драйвера
PROVIDER	COM-поставщик. Необязательный атрибут. Только для OLE DB
PROVIDERDSN	Имя поставщика данных. Необязательный атрибут. Только для OLE DB
DEBUG	Используется для отладки запроса. Необязательный атрибут. Возможные значения: • Yes — да; • No — нет (по умолчанию)

Тег `<CFQUERY>` возвращает сведения о запросе. В частности конструкция `CFQUERY.ExecutionTime` возвращает измеряемое в миллисекундах время, которое требуется запросу для его выполнения.

Так же тег `<CFQUERY>` создает объект запроса, обеспечивающий доступ к вспомогательной информации, посредством следующих переменных:

- ❑ `queryName.recordCount` — количество записей, полученных в результате запроса;
- ❑ `queryName.currentRow` — текущая запись запроса при использовании тега `<CFOUTPUT>`;
- ❑ `queryName.columnList` — список колонок запроса, перечисленных через запятую.

Прежде чем демонстрировать какие-либо примеры, рассмотрим синтаксис SQL-команды `SELECT`, позволяющей извлекать данные из одной или нескольких таблиц базы данных.

Команда **SELECT**

Предложение `SELECT` задает поля, константы и выражения, которые будут отображены в результатах запроса.

Синтаксис:

```
SELECT [ALL | DISTINCT]
       select_item [AS column_name]
       [, select_item [AS column_name] ...]
FROM table_name [, table_name ...]
[WHERE join_condition [AND join_condition ...]
  [AND |OR filter_condition [AND | OR filter_condition ...]]]
[GROUP BY group_column [, group_column ...]]
[HAVING filter_condition] [UNION [ALL] SELECTcommand]
[ORDER BY order_item [ASC | DESC] [, order_item [ASC | DESC] ...]]
```

Параметры:

- ❑ `[ALL | DISTINCT] select_item [AS column_name] [, select_item [AS column_name] ...]`

Опция `ALL` по умолчанию отображает в результатах запроса все строки.

Опция `DISTINCT` исключает из результатов запроса повторяющиеся строки. Ее указание в предложении `SELECT` допустимо только один раз.

Значение `select_item` задает элемент, который следует включить в столбец результатов запроса. Выбираемые элементы могут быть следующими:

- имя поля из таблицы, указанной в предложении `FROM`;
- константа, которая задает одинаковое постоянное значение во всех строках результатов запроса;
- выражение, которое может быть именем пользовательской функции.

Каждый элемент `select_item` генерирует один столбец результатов запроса. Если несколько элементов имеют одинаковые имена, укажите перед именем элемента псевдоним таблицы с точкой, чтобы столбцы не повторялись.

Замечание

Применение пользовательских функций в предложении `SELECT` обладает очевидными плюсами, но, тем не менее, необходимо иметь в виду, что скорость выполнения операций с помощью конструкции `SELECT` может быть ограничена скоростью выполнения этих пользовательских функций.

Для выбираемого элемента `SELECT`, являющегося полем или выражением с участием поля, могут быть использованы следующие функции полей:

- `AVG(select_item)` — вычисление среднего арифметического в столбце числовых данных;
- `COUNT(select_item)` — подсчет числа выбираемых элементов в столбце. `COUNT(*)` подсчитывает число строк в выводе запроса;
- `MIN(select_item)` — определение наименьшего значения среди выбираемых элементов `select_item` в столбце;
- `MAX(select_item)` — определение наибольшего значения `select_item` в столбце;
- `SUM(select_item)` — вычисление суммы по столбцу числовых данных.

❑ `AS column_name`

Задаёт заголовок столбца в выводе запроса. Это предложение полезно в случае, когда `select_item` является выражением или содержит функцию поля, а вы хотите назначить столбцу осмысленное имя. Имя `column_name` может быть выражением, но не может содержать символы, запрещённые в именах табличных полей.

❑ `FROM table_name [, table_name ...]`

Задаёт список таблиц, которые содержат данные, извлекаемые по запросу.

❑ `[WHERE join_condition [AND join_condition ...]`

`[AND | OR filter_condition [AND | OR filter_condition ...]]]`

Указывает, что в результат запроса включаются лишь некоторые записи. Предложение `WHERE` требуется при извлечении данных из нескольких таблиц.

Условие `join_condition` задаёт поля, которые связывают таблицы, указанные в предложении `FROM`. Если в запрос включено более одной таблицы, то для каждой таблицы, кроме первой, необходимо определить условие объединения.

Чтобы связать несколько условий объединения, воспользуйтесь оператором `AND`. Каждое условие объединения имеет следующий формат:

```
field_name1 comparison field_name2
```

где `field_name1` — имя поля одной таблицы; `field_name2` — имя поля другой таблицы, а `comparison` представляет один из операторов, представленных в табл. 11.2.

Таблица 11.2. Операторы условия объединения

Оператор	Тип сравнения
<code>=</code>	Равно
<code>==</code>	В точности равно
<code>LIKE</code>	Похоже
<code>< >, !=</code>	Не равно
<code>></code>	Больше
<code>> =</code>	Больше или равно
<code><</code>	Меньше
<code>< =</code>	Меньше или равно

❑ `[GROUP BY group_column [, group_column ...]]`

Группирует строки в запросе, исходя из значений в одном или нескольких столбцах. В качестве `group_column` можно задать имя обычного поля таблицы, имя поля, содержащего функцию поля SQL, или числовое выражение, указывающее позицию данного столбца в таблице результатов (самый левый столбец имеет номер 1).

❑ `[HAVING filter_condition]`

Задаёт условие фильтра, которому должны удовлетворять группы, чтобы быть включёнными в результаты запроса. Предложение `HAVING` следует использовать вместе с конструкцией `GROUP BY`. Оно может включать сколько угодно условий фильтра, связанных операторами `AND` и `OR`. Также можно использовать оператор `NOT`, изменяющий значение логического выражения на противоположное.

Условие `filter_condition` не может содержать подзапрос.

Предложение `HAVING` без предложения `GROUP BY` действует, как предложение `WHERE`.

❑ `[UNION [ALL] SELECTCommand]`

Комбинирует окончательные результаты одного запроса `SELECT` с окончательными результатами другого запроса `SELECT`. По умолчанию предло-

жение `UNION` проверяет скомбинированные результаты и удаляет повторяющиеся строки. Чтобы скомбинировать несколько предложений `UNION`, используйте скобки.

Если задана опция `ALL`, предложение `UNION` не будет удалять из комбинированных результатов повторяющиеся строки.

Предложения `UNION` подчиняются следующим правилам:

- с помощью этого предложения нельзя комбинировать подзапросы;
- у обеих команд `SELECT` в выводе запросов должно быть одинаковое число столбцов;
- каждый столбец результатов запроса одной команды `SELECT` должен иметь тот же тип данных и ту же ширину, что и соответствующий столбец другого запроса `SELECT`;
- только в окончательном варианте `SELECT` может присутствовать предложение `ORDER BY`, которое должно ссылаться на столбцы вывода по номерам. Если предложение `ORDER BY` включено, оно влияет на весь результат.

Предложение `UNION` также можно использовать для моделирования внешнего объединения (`outer join`).

Когда вы объединяете две таблицы в одном запросе, в вывод включаются только записи с совпадающими значениями в объединяемых полях. Если у некоторой записи родительской таблицы нет соответствующей ей записи в дочерней таблице, эта родительская запись не включается в вывод. Внешнее объединение позволяет включить в вывод все записи родительской таблицы вместе с совпадающими записями из дочерней таблицы.

❑ `[ORDER BY order_item [ASC | DESC] [, order_item [ASC | DESC] ...]]`

Сортирует результаты запроса по содержанию одного или нескольких столбцов. Каждый аргумент `order_item` должен соответствовать некоторому столбцу результатов и может представлять собой один из следующих элементов:

- поле в таблице, указанной в предложении `FROM`, которое также является выбираемым элементом в основном предложении `SELECT` (не в подзапросе);
- числовое выражение, указывающее местоположение столбца в таблице результатов. (Самый левый столбец имеет номер 1.)

Опция `ASC` задает возрастающий порядок сортировки результатов запроса, в соответствии с элементом (или элементами) `order_item`, и используется по умолчанию в предложении `ORDER BY`.

Опция `DESC` задает убывающий порядок сортировки результатов запроса.

Если вы не задали порядок результатов запроса с помощью предложения `ORDER BY`, они останутся неупорядоченными.

Первый запрос

Предположим, нам необходимо организовать предварительный запрос для открытия сеанса по заказу билетов на существующие круизы.

Напомним, в девятом пункте бизнес-правил (см. разд. "Бизнес-правила" гл. 5) мы определили, что при регистрации электронный адрес в то же время является входным именем пользователя (Login ID). В этом случае требуется создать диалоговую форму для ввода электронного адреса и пароля потенциального путешественника для последующей проверки наличия регистрации данного покупателя. Исходя из структуры базы данных CRUISE, непосредственно сам запрос будет выглядеть следующим образом:

```
<CFQUERY NAME="VerificationTra" DATASOURCE="#application.dsn#">
    SELECT key_tra, FirstName_tra, LastName_tra, EMail_tra, Gender_tra
    FROM Traveler
    WHERE EMail_tra = '#Form.EMail#'
           AND Password_tra = '#Form.Password#'
</CFQUERY>
```

где Form.Email и Form.Password — переменные формы, предварительно заполненные пользователем.

Следующим этапом мы можем создать переменные сессии, позволяющие в дальнейшем определять пользователя на протяжении всей текущей сессии и не затруднять его повторном заполнении диалоговой формы. Этот код выглядит так:

```
<CFIF VerificationTra.RecordCount eq 1>
    <CFSET Session.login = "true">
    <CFSET Session.traveler.id = #VerificationTra.key_tra#>
    <CFSET Session.traveler.gender = #VerificationTra.Gender_tra#>
    <CFSET Session.traveler.name = #VerificationTra.FirstName_tra# & " " &
                                   #VerificationTra.LastName_tra#>
    <CFINCLUDE TEMPLATE = "Welcome.cfm">
<CFELSE>
    <!-- Неудачная аутентификация (Authenticate failure) --->
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=BuyTickets.cfm?error=1">
</CFIF>
```

Смысл представленного кода заключается в определении количества записей, полученных в результате запроса, и т. к. разные покупатели не могут иметь один и тот же электронный адрес, то и количество записей данного запроса может равняться 1 при удачной проверке и 0 при неудачной проверке. В данном коде мы использовали встроенную переменную запроса RecordCount.

На рис. 11.1 показана форма для открытия сессии уже зарегистрированного покупателя в рамках проекта "My-Line Cruises".

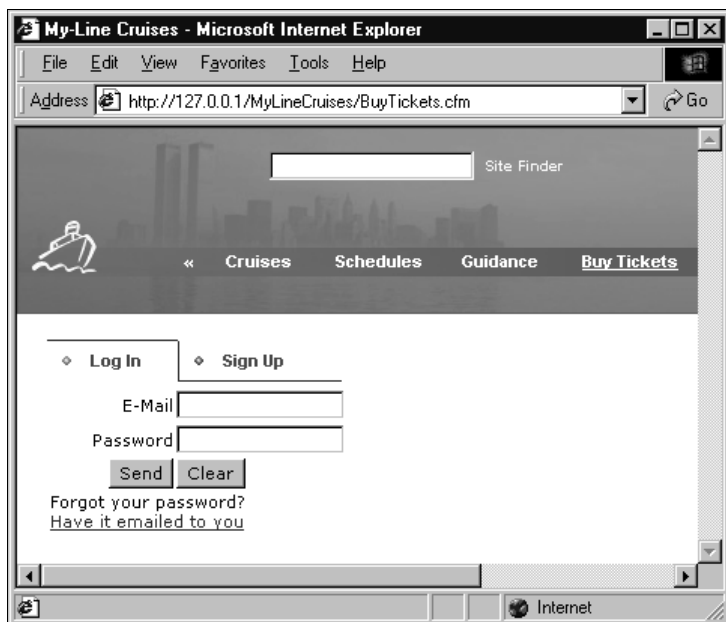


Рис. 11.1. Пример формы открытия сессии покупателя

Далее продемонстрируем полный код страницы BuyTickets.cfm, позволяющей решить поставленную перед нами задачу (листинг 11.1).

Листинг 11.1. Код страницы BuyTickets.cfm

```
<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <style type="text/css">
    <!--
      #LogIn { position:absolute; left:20; top:140; width:200;
        height:100; margin:0;z-index:0;}
    -->
  </style>
```

```

<script language="JavaScript">
  if (ie) {document.write('<link rel="stylesheet" type="text/css"
    href="main_ie.css" title="master">');}
function SendPassword() {
  if(ie) var EMail = document.form1.EMail.value;
  else var EMail =
    document.layers['LogIn'].document.form1.EMail.value;
  if (EMail=="") {
    alert('Enter please E-mail');
  } else {
    location.href="Sendpassword.cfm?EMail_tra="+EMail;
  }
}
</script>
</HEAD>
<BODY onload="init(1)"
  bgcolor="#<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
  text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<CFIF isDefined("Session.traveler.id")>
  <!-- The traveler is already determined -->
  <CFINCLUDE TEMPLATE = "Welcome.cfm">
<CFELSE>
  <CFIF isDefined("Form.EMail") AND isDefined("Form.Password")>
    <!-- Проверка путешественника (Verification traveler) -->
    <CFQUERY NAME="VerificationTra" DATASOURCE="#application.dsn#">
      SELECT Key_tra, FirstName_tra, LastName_tra, EMail_tra,
        Gender_tra
      FROM Traveler
      WHERE EMail_tra = '#Form.EMail#'
        AND Password_tra = '#Form.Password#'
    </CFQUERY>
    <CFIF VerificationTra.RecordCount eq 1>
      <CFSET Session.login = "true">
      <CFSET Session.traveler.id = #VerificationTra.key_tra#>
      <CFSET Session.traveler.gender = #VerificationTra.Gender_tra#>
      <CFSET Session.traveler.name = #VerificationTra.FirstName_tra#
        & " " & #VerificationTra.LastName_tra#>

```

```

        <CFINCLUDE TEMPLATE = "Welcome.cfm">
    <CFELSE>
        <!--- Неудачная аутентификация (Authenticate failure) --->
        <META HTTP-EQUIV="Refresh" CONTENT="0;
            URL=BuyTickets.cfm?error=1">
    </CFIF>
<CFELSE>
    <CFIF isDefined("Error")>
        <CFSET msgError="#application.msgErrorLogin#">
    <CFELSE>
        <CFSET msgError="">
    </CFIF>
    <DIV id="LogIn">
    <FORM action='BuyTickets.cfm' method='post' name='form1'>
        <TABLE border='0' cellspacing='0' cellpadding='0'>
            <TR>
                <TR><TD colspan="2" bgcolor="#122c6d" height=1></TD>
                    <TD colspan="2" height=1></TD></TR>
                <TD class="menleft">
                    <IMG name="bull1" src="bullet2.gif" width="20"
                        height="20" border="0"></TD>
                <TD width="55" class="menleft">Log In</TD>
                <TD bgcolor="#122c6d"></TD>
                <TD class="menleft"><a href="SignUp.cfm"
                    onMouseOver="mouse_over_bull('bull2', 'LogIn');
                    window.status='Sign Up'; return true"
                    onMouseOut="mouse_out_bull('bull2', 'LogIn');
                    window.status=''; return true">
                    <IMG name="bull2" src="bullet.gif" width="20" height="20"
                        border="0"></a></TD>
                <TD width="82"><a href="SignUp.cfm"
                    onMouseOver="mouse_over_bull('bull2', 'LogIn');"
                    onMouseOut="mouse_out_bull('bull2', 'LogIn');"
                    class="menleft">Sign Up</a></TD>
            <TR><TD colspan="2" height=1></TD>
                <TD colspan="4" bgcolor="#122c6d" height=1></TD></TR>
        </TR>
    </TABLE>
    <TABLE border='0' cellspacing='2' cellpadding='0'>

```

```

<TR><TD colspan="2" align="center" class="txtError">
    <CFOUTPUT>#msgError#</CFOUTPUT></td></tr>
<TR><TD width="100" align="right" class="txtForm1">
    E-Mail</td><td class="note">
        <input name='EMail' size='16'></td></tr>
<TR><TD align="right" class="txtForm1">
    Password</td><td class="note">
        <input type='password' name='Password'
            size='16'></td></tr>
<TR><TD align="right">
        <input type='submit' value='Send'></td><td>
        <input type='reset' value='Clear'></td></tr>
<TR><TD colspan="2" align="left" class="txtForm1">
    Forgot your password?<br>
    <a href="javascript:SendPassword();"
        onMouseOver="window.status='Have it emailed to you';
        return true"
        onMouseOut="window.status=''; return true">
        Have it emailed to you</a></td></tr>
</TABLE>
</FORM>
</DIV>
</CFIF>
</CFIF>
</BODY>
</HTML>

```

На рис. 11.2 показан результат удачного открытия сессии покупателя.

Как видно из примера, в результате удачного открытия сеанса мы включаем с помощью тега <CFINCLUDE> страницу Welcome.cfm. В листинге 11.2 продемонстрируем код страницы Welcome.cfm.

Листинг 11.2. Код страницы Welcome.cfm

```

<CFIF #Session.traveler.gender# eq "1">
    <CFSET Appeal="Ms.">
<CFELSE>
    <CFSET Appeal="Mr.">
</CFIF>

```

```

<CFOUTPUT>

<div id="LogIn">
    <p class="smalltxt">Welcome<br>Dear #Appeal#
    <a href="UpdateTra.cfm">#Session.traveler.name#</a>
        , thank you for your interest in My-Cruises Line.</p>
</div>
</CFOUTPUT>

<!-- The information block on cruises -->
<!-- ... -->

```

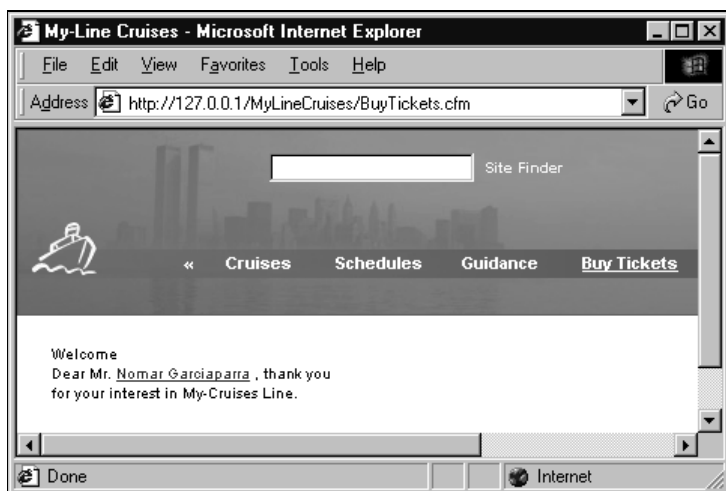


Рис. 11.2. Результат удачного открытия сессии

В нашем примере именно на странице `Welcome.cfm` подразумевается размещение информации о текущих круизах. Обращаем ваше внимание на то, что в данном примере встречается ссылка на страницу `UpdateTra.cfm`, предназначенную для обновления данных о пользователе текущей сессии. Однако об этом чуть позже.

Отметим лишь, что для удачного открытия сессии необходимо наличие в базе данных сведений о зарегистрированных пользователях. Далее мы продемонстрируем пример добавления данных с помощью ColdFusion.

Добавление данных

Для добавления данных в ColdFusion можно прибегнуть к нескольким способам, например, указать команду внутри уже описанного нами тега `<CFQUERY>`, применить тег `<CFINSERT>` или тег `<CFGRID>`, частично исполь-

зуемого в рамках тега `<CFFORM>`. И если о последнем способе мы поговорим в следующей главе, то первые два варианта рассмотрим прямо сейчас.

Прежде чем привести пример добавления данных, рассмотрим синтаксис SQL-команды `INSERT`.

Команда *INSERT*

Синтаксис:

```
INSERT INTO table_name [(fname1 [, fname2, ...])]
VALUES (expression1 [, expression2, ...])
```

Параметры:

❑ `INSERT INTO table_name`

Задаёт имя таблицы, к которой добавляется новая запись.

❑ `[(fname1 [, fname2 [, ...]])]`

Определяет имена полей новой записи, в которые вставляются значения.

❑ `VALUES (expression1 [, expression2 [, ...]])`

Задаёт значения полей, вставляемые в новую запись. Если имена полей опущены, нужно указывать значения полей в порядке, определяемом структурой таблицы.

Пример команды *INSERT*

Допустим, нам необходимо организовать регистрацию потенциальных покупателей билетов на существующие круизы.

В этом случае требуется создать диалоговую форму для ввода необходимых данных для последующего их добавления в базу данных `CRUISE`. Тогда запрос в ColdFusion будет выглядеть следующим образом:

```
<CFSET Birthday = #DateFormat(#Form.Birthday#, "yyyy-mm-dd")#>
<CFQUERY NAME="AddTraveler" DATASOURCE="#application.dsn#">
    INSERT INTO Traveler(EMail_tra, FirstName_tra,
                        LastName_tra, MiddleInitial_tra,
                        Password_tra, Birthday_tra,
                        PhoneNum_tra, Gender_tra)
    VALUES ('#Form.EMail#', '#Form.FirstName#',
            '#Form.LastName#', '#Form.MiddleInitial#',
            '#Form.Password#', '#Form.Birthday#',
            '#Form.PhoneNum#', #Form.Gender#)
</CFQUERY>
```


Здесь в предложении `VALUES` перечислены переменные формы, предварительно заполненные пользователем.

Следует отметить, что мы используем БД Sybase, где формат даты по умолчанию представляется как "уууу-мм-дд", в свою очередь при вводе данных пользователю предлагается вводить свой день рождения с маской "дд-мм-уууу". Поэтому для приведения формата даты к общему знаменателю используется существующая для этих целей функция `DateFormat`.

Далее будем действовать по аналогии с примером открытия сессии, рассмотренным чуть ранее, где создавали переменные сессии, позволяющие в дальнейшем идентифицировать пользователя на протяжении всей текущей сессии и не затруднять его повторным заполнением диалоговой формы. Этот код выглядит так:

```
<CFQUERY NAME="TravelerExtKey" DATASOURCE="#application.dsn#">
    SELECT key_tra
        FROM Traveler
        WHERE EMail_tra='#Form.EMail#'
</CFQUERY>

<CFOUTPUT QUERY="TravelerExtKey">
    <CFSET Session.login = "true">
    <CFSET Session.traveler.id = #TravelerExtKey.key_tra#>
    <CFSET Session.traveler.gender = #Form.Gender#>
    <CFSET Session.traveler.name = #Form.FirstName# & " " &
        #Form.LastName#>
</CFOUTPUT>
```

Как видно из примера, прежде чем определить переменную идентификатора пользователя, нам пришлось создать дополнительный запрос на выборку по полю `key_tra`, т. к. в базе данных `CRUISE` все уникальные ключи присваиваются автоматически. И только с помощью обращения к уже добавленной записи мы определяем этот уникальный ключ.

На рис. 11.3 показана форма для регистрации покупателя, в рамках проекта "My-Line Cruises".

В разд. "Проверка переменных" гл. 9 мы уже рассматривали похожий пример по регистрации пользователей без применения базы данных. Здесь же помимо добавления блоков по работе с базой данных мы также несколько изменили код в соответствии с требованием проекта "My-Line Cruises".

Далее продемонстрируем полный код страницы `SignUp.cfm`, позволяющей решить поставленную перед нами задачу (листинг 11.3).

My-Line Cruises - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://127.0.0.1/MyLineCruises/SignUp.cfm> Go

Site Finder

« Cruises Schedules Guidance Buy Tickets

♦ Log In ♦ Sign Up

E-Mail (Login) *

Password *

Confirm *

First Name *

Last Name *

Middle Initial

Birthday dd-mm-yyyy

Phone Num.

Gender ▼

Internet

Рис. 11.3. Пример формы регистрации покупателя

Листинг 11.3. Код страницы SignUp.cfm

```

<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <style type="text/css">
    <!--
      #SignUp { position:absolute; left:20; top:140; width:200;
        height:100; margin:0;z-index:0;}
    -->
  </style>

```

```

<script language="JavaScript">
    if (ie) {document.write('<link rel="stylesheet" type="text/css"
                                href="main_ie.css" title="master">');}

</script>
</HEAD>
<BODY onload="init(1)"
    bgcolor="#<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
    text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<CFIF isDefined("Session.traveler.id")>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=BuyTickets.cfm">
<CFELSE>
    <CFIF isDefined("Form.Email")>
        <CFSET Disuccessful = 1>
        <CFSET msgError = "Warning Error!">
        <CFIF Len(Form.Email) le 5 and Form.Email does not contain "@">
            <CFSET msgError = msgError & "<br>" & #application.msgError1#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Len(Form.Password) le 4>
            <CFSET msgError = msgError & "<br>" & #application.msgError2#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF IsNumeric(Form.Password)>
            <CFSET msgError = msgError & "<br>" & #application.msgError3#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Form.Password not equal Form.Confirm>
            <CFSET msgError = msgError & "<br>" & #application.msgError4#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Len(Form.FirstName) le 0>
            <CFSET msgError = msgError & "<br>" & #application.msgError5#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Len(Form.LastName) le 0>
            <CFSET msgError = msgError & "<br>" & #application.msgError6#>
            <CFSET Disuccessful = 0>
        </CFIF>

```

```
<CFIF not IsDate(Form.Birthday) and Len(Form.Birthday)gt 0>
  <CFSET msgError = msgError & "<br>" & #application.msgError7#>
  <CFSET Disuccessful = 0>
</CFIF>

<CFIF YesNoFormat(Disuccessful) equal "No"> <!-- Errors -->
  <CFSET email = #Form.Email#>
  <CFSET fname = #Form.FirstName#>
  <CFSET lname = #Form.LastName#>
  <CFSET middle = #Form.MiddleInitial#>
  <CFSET bday = #Form.Birthday#>
  <CFSET phonen = #Form.PhoneNum#>
<CFELSE>
  <!-- Проверка путешественника (Verification traveler) --->
  <CFQUERY NAME="VerificationSU" DATASOURCE="#application.dsn#">
    SELECT key_tra
      FROM Traveler
      WHERE Email_tra = '#Form.Email#'
  </CFQUERY>
  <CFIF VerificationSU.RecordCount eq 1>
    <!-- The record with this Email address already exists! --->
  <CFELSE>
    <!-- Добавление записи (Add record) --->
    <CFSET Birthday = #DateFormat(#Form.Birthday#,
                                   "yyyy-mm-dd") #>
    <CFQUERY NAME="AddTraveler" DATASOURCE="#application.dsn#">
      INSERT INTO Traveler(Email_tra, FirstName_tra,
                           LastName_tra, MiddleInitial_tra,
                           Password_tra, Birthday_tra,
                           PhoneNum_tra, Gender_tra)
        VALUES('#Form.Email#', '#Form.FirstName#',
                 '#Form.LastName#', '#Form.MiddleInitial#',
                 '#Form.Password#', '#Form.Birthday#',
                 '#Form.PhoneNum#', #Form.Gender#)
    </CFQUERY>
    <!-- Извлечение по ключу (Extraction of a key) --->
    <CFQUERY NAME="TravelerExtKey"
      DATASOURCE="#application.dsn#">
      SELECT key_tra
```

```

        FROM Traveler WHERE EMail_tra='#Form.Email#'
    </CFQUERY>
    <CFOUTPUT QUERY="TravelerExtKey">
        <CFSET Session.login = "true">
        <CFSET Session.traveler.id = #TravelerExtKey.key_tra#>
        <CFSET Session.traveler.gender = #Form.Gender#>
        <CFSET Session.traveler.name = #Form.FirstName# & " " &
            #Form.LastName#>
    </CFOUTPUT>
</CFIF>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=BuyTickets.cfm">
<CFEXIT>
</CFIF>
<CFELSE>
    <CFSET email = "">
    <CFSET fname = "">
    <CFSET lname = "">
    <CFSET middle = "">
    <CFSET bday = "">
    <CFSET phonen = "">
    <CFSET msgError = "">
</CFIF>
<DIV id="SignUp">
    <FORM action='SignUp.cfm' method='post' name='form1'>
        <TABLE border='0' cellspacing='0' cellpadding='0'>
            <TR>
                <TR><TD colspan="2" height=1></TD>
                    <TD colspan="4" bgcolor="#122c6d" height=1></TD></TR>
                <TD class="menleft"><a href="BuyTickets.cfm"
                    onMouseOver="mouse_over_bull('bull12', 'SignUp');
                    window.status='Sign Up'; return true"
                    onMouseOut="mouse_out_bull('bull12', 'SignUp');
                    window.status=''; return true">
                    <IMG name="bull12" src="bullet.gif" width="20"
                        height="20" border="0"></a></TD>
                <TD width="55"><a href="BuyTickets.cfm"
                    onMouseOver="mouse_over_bull('bull12', 'SignUp');"
                    onMouseOut="mouse_out_bull('bull12', 'SignUp');"
                    class="menleft">Log In</a></TD>

```

```

<TD bgcolor="#122c6d"></TD>
<TD class="menleft">
    <IMG name="bull1" src="bullet2.gif" width="20"
        height="20" border="0"></TD>
<TD width="82" class="menleft">Sign Up</TD>
<TR><TD colspan="2" bgcolor="#122c6d" height=1>
    </TD><TD colspan="2" height=1></TD></TR>
</TR>
</TABLE>
<TABLE width="300" border="0" cellspacing="2" cellpadding="0">
<TR><TD colspan="2" align="center" class="txtError">
    <CFOUTPUT>#msgError#</CFOUTPUT></TD></TR>
<TR><TD align="right" class="txtForm1">
    E-Mail (Login)</td><td width="215" class="txtForm1">
    <input name='EMail' size='16'
        value="<CFOUTPUT>#email#</CFOUTPUT>">&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
    Password</td><td class="txtForm1">
    <input type='password' name='Password'
        size='16'>&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
    Confirm</td><td class="txtForm1">
    <input type='password' name='Confirm'
        size='16'>&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
    First Name</td><td class="txtForm1">
    <input name='FirstName' size='16'
        value="<CFOUTPUT>#fname#</CFOUTPUT>">&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
    Last Name</td><td class="txtForm1">
    <input name='LastName' size='16'
        value="<CFOUTPUT>#lname#</CFOUTPUT>">&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
    Middle Initial</td><td>
    <input name='MiddleInitial' size='1'
        value="<CFOUTPUT>#middle#</CFOUTPUT>"></TD></TR>

```

```

<TR><TD align="right" class="txtForm1">
    Birthday</td><td class="txtForm1">
        <input name='Birthday' size='10'
            value="<CFOUTPUT>#bday#</CFOUTPUT>">&nbsp;
            dd-mm-yyyy</TD></TR>
<TR><TD align="right" class="txtForm1">
    Phone Num.</td><td>
        <input name='PhoneNum' size='10'
            value="<CFOUTPUT>#phonen#</CFOUTPUT>"></TD></TR>
<TR><TD align="right" class="txtForm1">
    Gender</TD><TD>
        <select name='Gender'>
            <option value="0">male
            <option value="1">female
        </select></TD></TR>
<TR><TD align="right">
        <input type='submit' value=' Add '></TD><TD>
        <input type='reset' value='Clear'></TD></TR>
</TABLE>
</FORM>
</DIV>
</CFIF>
</BODY>
</HTML>

```

Отметим то, что сообщения об ошибках определены в конфигурационном файле Application.cfm так, как это показано в *разд. "Пример файла Application.cfm" гл. 10*.

Также для добавления данных в ColdFusion можно использовать тег `<CFINSERT>`, о чем и пойдет речь в следующем разделе с предварительным описанием этого тега.

Тег `<CFINSERT>`

Синтаксис:

```

<CFINSERT DATASOURCE = "ds_name"
    DBTYPE = "type"
    DBSERVER = "dbms"
    DBNAME = "database name"
    TABLENAME = "tbl_name"

```

```

CONNECTSTRING = "connection string"
TABLEOWNER = "owner"
TABLEQUALIFIER = "tbl_qualifier"
USERNAME = "username"
PASSWORD = "password"
PROVIDER = "COMProvider"
PROVIDERDSN = "datasource"
FORMFIELDS = "formfield1, formfield2, ...">

```

В табл. 11.3 представлено описание атрибутов тега <CFINSERT>.

Таблица 11.3. Описание атрибутов тега <CFINSERT>

Атрибут	Описание
DATASOURCE	Источник данных, к которому обращен запрос. Обязательный атрибут для всех DBTYPE кроме DBTYPE="query" и DBTYPE="dynamic"
DBTYPE	Тип драйвера базы данных. Необязательный атрибут. Возможные значения: • <code>dynamic</code> — тип соединения с источником данных ODBC, который не определен в "ColdFusion Administrator". При использовании этого значения следует определить всю информацию о связи ODBC с атрибутом <code>CONNECTSTRING</code>; • <code>query</code> — использует существующий запрос как источник данных. В этом случае нет необходимости определять атрибут <code>DATASOURCE</code>, вместо этого следует использовать имя существующего запроса в качестве имени таблицы; • <code>ODBC</code> — ODBC-драйвер (по умолчанию); • <code>Oracle73</code> — позволяет использовать собственный драйвер БД Oracle 7.3. В этом случае на компьютере, где установлен ColdFusion Server, должна быть установлена клиентская часть Oracle 7.3.4.0.0 (или более поздняя версия); • <code>Oracle80</code> — дает возможность использовать собственный драйвер БД Oracle 8.0. Соответственно должна быть установлена клиентская часть Oracle 8.0 (или более поздняя версия); • <code>Sybase11</code> — позволяет использовать собственный драйвер БД Sybase System 11. Соответственно должна быть установлена клиентская часть Sybase System 11.1.1 (или более поздняя версия); рекомендуется с исправлениями ebf 7729; • <code>OLEDB</code> — поставщик OLE DB; • <code>DB2</code> — использование собственного драйвера БД DB2 5.2; • <code>Informix73</code> — использование собственного драйвера БД Informix 7.3

Таблица 11.3 (продолжение)

Атрибут	Описание
DBSERVER	Имя сервера БД. Необязательный атрибут. Применяется для собственных драйверов и SQL OLEDB. При определении DBSERVER игнорируется установка источника данных
DBNAME	Имя БД (только для Sybase System 11 и SQL OLEDB). Необязательный атрибут. При определении DBNAME игнорируется БД по умолчанию, установленная в источнике данных
TABLENAME	Имя таблицы. Обязательный атрибут. Обратите внимание на следующее: • при использовании драйверов Oracle значение этого атрибута должно быть записано в верхнем регистре; • при использовании драйвера Sybase значение этого атрибута должно быть в том же самом регистре, в котором была создана данная таблица.
CONNECTSTRING	Строка соединения для отправления к серверу ODBC. Требуется для DBTYPE="dynamic". Необязательный атрибут для всех других типов. При соединении с источником данных, определенным в "ColdFusion Administrator", можно использовать этот атрибут, чтобы определить дополнительные детали соединения или переопределить информацию соединения. Если вы динамически соединяетесь с источником данных, указанным в атрибуте DATASOURCE, определяя DBTYPE="dynamic", строка соединения должна определить все требуемые параметры соединения ODBC
TABLEOWNER	Владелец таблицы. Необязательный атрибут. Может применяться для источников данных, поддерживающих собственность (ownership) на таблицу (например, SQL Server, Oracle и Sybase SQL Anywhere)
TABLEQUALIFIER	Описатель таблицы. Необязательный атрибут. Может использоваться для источников данных, поддерживающих описатель таблиц (qualifiers). Для SQL Server и Oracle обращается к имени базы данных, которая содержит таблицу. Для dBase-драйверов происходит обращение к каталогу, где размещаются dbf-файлы
USERNAME	Имя пользователя. Необязательный атрибут. При определении данного атрибута значение USERNAME, указанное в источнике данных, игнорируется
PASSWORD	Пароль пользователя. Необязательный атрибут. При определении данного атрибута значение PASSWORD, указанное в источнике данных, игнорируется
PROVIDER	COM-поставщик. Необязательный атрибут. Только для OLE DB

Таблица 11.3 (окончание)

Атрибут	Описание
PROVIDERDSN	Имя поставщика данных. Необязательный атрибут. Только для OLE DB
FORMFIELDS	Список полей, разделенных запятыми. Необязательный атрибут. Если этот атрибут не определен, то все поля формы участвуют в операции добавления

Пример тега <CFINSERT>

Чтобы демонстрировать использование тега <CFINSERT>, предназначенного для добавления записей в базу данных, рассмотрим пример, позволяющий регистрировать новые экскурсионные поездки и круизы.

Решим данную задачу по аналогии с предыдущим примером: создадим диалоговую форму, предназначенную для ввода необходимых данных с целью их последующего добавления в базу данных CRUISE, а именно — в таблицу TOUR. Прежде чем представить полный код страницы, продемонстрируем отрывок этого кода, где, собственно говоря, и происходит добавление новых значений:

```
<!-- Проверка типа (Verification tour) -->
<CFQUERY NAME="VerificationNewTour" DATASOURCE="#application.dsn#">
    SELECT key_tou
        FROM Tour WHERE Name_tou='#Form.Name_tou#'
</CFQUERY>
<CFIF VerificationNewTour.RecordCount eq 1>
    <!-- The record with this Name tour already exists! -->
<CFELSE>
    <!-- Добавление записи (Add record) -->
    <CFINSERT DATASOURCE = "#application.dsn#"
        TABLENAME = "Tour"
        FORMFIELDS = "Name_tou,PointDepart_tou,NumLandingPlaces_tou,
            Time_tou,CostAdult_tou,CostChildren_tou,
            CostSenior_tou,Photo_tou,Note_tou">
</CFIF>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=AdminListTour.cfm">
```

Здесь, прежде чем произвести добавление новой записи, создаем дополнительный запрос, позволяющий проверить, существует ли данный круиз в текущей базе данных. И только при отсутствии круиза совершаем добавле-

ние введенных пользователем данных и активизируем страницу со списком всех зарегистрированных круизов.

На рис. 11.4 показана форма для регистрации круизов в рамках проекта "My-Line Cruises".

Рис. 11.4. Пример формы регистрации круизов

В листинге 11.4 представлен полный код страницы AdminNewTour.cfm, позволяющей решить сформулированную выше задачу.

Листинг 11.4. Код страницы AdminNewTour.cfm

```
<HTML>
<HEAD>
    <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
```

```

<link rel="stylesheet" type="text/css" href="main.css">
<script type="text/javascript" language="JavaScript"
    src="main.js"></script>
<style type="text/css">
<!--
#Tour { position:absolute; left:20; top:140; width:200; height:100;
    margin:0; z-index:0;}
-->
</style>
<script language="JavaScript">
    if (ie) {document.write('<link rel="stylesheet" type="text/css"
        href="main_ie.css" title="master">');}
</script>
</HEAD>
<BODY onload="init(1)"
    bgcolor="#CFOUTPUT">#application.BGColorPage#</CFOUTPUT>
    text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<CFIF isDefined("Form.Name_tou")>
    <CFSET Disuccessful = 1>
    <CFSET msgError = "Warning Error!">
<CFIF Len(Form.Name_tou) le 0 or Len(Form.PointDepart_tou) le 0 or
    Len(Form.NumLandingPlaces_tou) le 0 or Len(Form.Time_tou) le 0 or
    Len(Form.CostAdult_tou) le 0 or Len(Form.CostChildren_tou) le 0
        or Len(Form.CostSenior_tou) le 0>
    <CFSET msgError = msgError & "<br>" & "The fields marked with
        a symbol *, should not be empty!">
    <CFSET Disuccessful = 0>
</CFIF>
<CFIF YesNoFormat(Disuccessful) equal "No"> <!-- Ошибки (Errors) -->
    <CFSET Name_tou = #Form.Name_tou#>
    <CFSET PointDepart_tou = #Form.PointDepart_tou#>
    <CFSET NumLandingPlaces_tou = #Form.NumLandingPlaces_tou#>
    <CFSET Time_tou = #Form.Time_tou#>
    <CFSET CostAdult_tou = #Form.CostAdult_tou#>
    <CFSET CostChildren_tou = #Form.CostChildren_tou#>
    <CFSET CostSenior_tou = #Form.CostSenior_tou#>
    <CFSET Photo_tou = #Form.Photo_tou#>
    <CFSET Note_tou = #Form.Note_tou#>

```

```

<CFELSE>
    <!--- Verification tour --->
    <CFQUERY NAME="VerificationNewTour"
        DATASOURCE="#application.dsn#"
        SELECT key_tou
            FROM Tour WHERE Name_tou = '#Form.Name_tou#'
    </CFQUERY>
    <CFIF VerificationNewTour.RecordCount eq 1>
        <!--- The record with this Name tour already exists! --->
    <CFELSE>
        <!--- Добавление записи (Add record) --->
        <CFINSERT DATASOURCE = "#application.dsn#"
            TABLENAME = "Tour"
            FORMFIELDS = "Name_tou,PointDepart_tou,
                NumLandingPlaces_tou,
                Time_tou,CostAdult_tou,
                CostChildren_tou,CostSenior_tou,
                Photo_tou,Note_tou">

    </CFIF>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=AdminListTour.cfm">
    <CFEXIT>
</CFIF>
<CFELSE>
    <CFSET Name_tou = "">
    <CFSET PointDepart_tou = "">
    <CFSET NumLandingPlaces_tou = "">
    <CFSET Time_tou = "">
    <CFSET CostAdult_tou = "">
    <CFSET CostChildren_tou = "">
    <CFSET CostSenior_tou = "">
    <CFSET Photo_tou = "">
    <CFSET Note_tou = "">
    <CFSET msgError = "">
</CFIF>
<DIV id="Tour">
    <FORM action='AdminNewTour.cfm' method='post' name='form1'>
        <TABLE width="370" border="0" cellspacing="0" cellpadding="0">
            <TR>
                <TR><TD colspan="2" height=1></td>

```

```

<TD colspan="4" bgcolor="#122c6d" height=1></TD></TR>
<TD class="menleft"><a href="AdminListTour.cfm"
    onMouseOver=" mouse_over_bull('bull2', 'Tour');
    window.status='List Tour'; return true"
    onMouseOut="mouse_out_bull('bull2', 'Tour');
    window.status=''; return true">
    </a></td>
<TD width="155"><a href="AdminListTour.cfm"
    onMouseOver=" mouse_over_bull('bull2', 'Tour');"
    onMouseOut=" mouse_out_bull('bull2', 'Tour');"
        class="menleft">
        List Tour</a></TD>
<TD bgcolor="#122c6d"></TD>
<TD class="menleft">
    </TD>
<td width="215" class="menleft">New Tour</TD>
<TR><TD colspan="2" bgcolor="#122c6d" height=1></TD>
    <TD colspan="2" height=1></TD></TR>
</TR>
</TABLE>
<TABLE width="390" border="0" cellspacing="2" cellpadding="0">
    <TR><TD colspan="2" align="center" class="txtError">
        <CFOUTPUT>#msgError#</CFOUTPUT></TD></TR>
    <TR><TD align="right" class="txtForm1">
        Name tour</td><td width="230" class="txtForm1">
        <input name='Name_tou' size='30'
            value="<CFOUTPUT>#Name_tou#</CFOUTPUT>">&nbsp;*</td></TR>
    <TR><TD align="right" class="txtForm1">
        Point of departure</td><td class="txtForm1">
        <input name='PointDepart_tou' size='30'
            value="<CFOUTPUT>#PointDepart_tou#</CFOUTPUT>">&nbsp;*</td></TR>
    <TR><TD align="right" class="txtForm1">
        Number of landing places</td><td class="txtForm1">
        <input name='NumLandingPlaces_tou' size='6'
            value="<CFOUTPUT>#NumLandingPlaces_tou#</CFOUTPUT>">&nbsp;*</td></TR>

```

```

</TD></TR>
<TR><TD align="right" class="txtForm1">
    Time in tour</TD><TD class="txtForm1">
        <input name='Time_tou' size='2'
            value="<CFOUTPUT>#Time_tou#</CFOUTPUT>">&nbsp;hours,&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    Cost for adults</td><td class="txtForm1">
        <input name='CostAdult_tou' size='6'
            value="<CFOUTPUT>#CostAdult_tou#</CFOUTPUT>">&nbsp;$,&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    Cost for children</td><td class="txtForm1">
        <input name='CostChildren_tou' size='6'
            value="<CFOUTPUT>#CostChildren_tou#</CFOUTPUT>">&nbsp;$,&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    Cost for senior citizens</td><td class="txtForm1">
        <input name='CostSenior_tou' size='6'
            value="<CFOUTPUT>#CostSenior_tou#</CFOUTPUT>">&nbsp;$,&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">Photo</TD><TD>
        <input name='Photo_tou' type="File" accept="image/gif"
            size='23' value="<CFOUTPUT>#Photo_tou#</CFOUTPUT>">
</TD></TR>
<TR><TD align="right" class="txtForm1">Note</TD><TD>
        <textarea name = "Note_tou" cols = "33" rows = "4">
            <CFOUTPUT>#Note_tou#</CFOUTPUT></textarea></TD></TR>
<TR><TD align="right">
        <input type='submit' value=' Add '></TD><TD>
        <input type='reset' value='Clear'></TD></TR>
</TABLE>
</FORM>
</DIV>
</BODY>
</HTML>

```

Обращаем ваше внимание на то, что в данном примере переменные формы именуются в соответствии с названием полей таблицы приемника данных.

Вывод результатов

Теперь, когда мы уже имеем представление о построении запросов и добавлении данных, видимо следует перейти к объяснению, каким образом можно выводить данные, полученные в результате выполнения того или иного запроса.

Итак, для вывода данных в ColdFusion можно прибегнуть к нескольким способам: либо использовать тег `<CFOUTPUT>`, либо `<CFTABLE>` в сочетании с тегом `<CFCOL>`. Впрочем, для этой цели также можно воспользоваться тегом `<CFGRID>`, используемым в рамках тега `<CFFORM>`, но об этом поговорим в следующей главе.

Тег `<CFOUTPUT>`

Синтаксис:

```
<CFOUTPUT QUERY = "query_name"
           GROUP = "query_column"
           GROUPCASESENSITIVE = "Yes | No"
           STARTROW = "start_row"
           MAXROWS = "max_rows_output">
</CFOUTPUT>
```

В табл. 11.4 представлено описание атрибутов тега `<CFOUTPUT>`.

Таблица 11.4. Описание атрибутов тега `<CFOUTPUT>`

Атрибут	Описание
QUERY	Имя запроса. Необязательный атрибут. Для вывода результатов запроса внутри блока <code><CFOUTPUT></code>
GROUP	Имя колонки запроса. Необязательный атрибут. Предполагает группировку результата запроса по указанной колонке (полю таблицы)
GROUPCASESENSITIVE	Восприимчивость к регистру. Необязательный атрибут. Используется для указания учета регистра при группировке результата запроса. Возможные значения: - Yes — да (по умолчанию); - No — нет
STARTROW	Номер записи, с которой следует выводить полученные данные. Необязательный атрибут
MAXROWS	Максимальное количество записей, которое следует выводить внутри блока <code><CFOUTPUT></code> . Необязательный атрибут

Пример тега <CFOUTPUT>

Мы уже неоднократно применяли тег <CFOUTPUT> для вывода значений различных переменных, но при этом ни разу не использовали данную конструкцию в сочетании с описанными выше атрибутами. Здесь же мы постараемся восполнить этот пробел.

Предположим, перед нами стоит задача вывести все круизы, зарегистрированные в базе данных.

Для решения данной задачи предварительно следует построить запрос, например таким образом:

```
<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
    SELECT Key_tou, Name_tou, PointDepart_tou, NumLandingPlaces_tou,
           Time_tou, CostAdult_tou, CostChildren_tou, CostSenior_tou
    FROM Tour
</CFQUERY>
```

где GetTour — имя запроса.

После чего выведем список полученных данных с указанием имени запроса следующим образом:

```
<TABLE width="630" border="0" cellspacing="2" cellpadding="0">
    <CFOUTPUT QUERY="GetTour">
        <TR align="right">
            <TD class="txtTab" align="left">
                <a href="AdminUpdateTour.cfm?key_tou=#Key_tou#"
                #Name_tou#</a></TD>
            <TD class="txtTab" align="left">#PointDepart_tou#</TD>
            <TD class="txtTab">#NumLandingPlaces_tou#</TD>
            <TD class="txtTab">#Time_tou#-hr</TD>
            <TD class="txtTab">$$CostAdult_tou#</TD>
            <TD class="txtTab">$$CostChildren_tou#</TD>
            <TD class="txtTab">$$CostSenior_tou#</TD>
            <TD class="txtTab" align="center">
                <a href="AdminNoteTour.cfm?key_tou=#Key_tou#" target="_blank">
                    <FONT face="Wingdings">±</FONT></a></TD>
        </TR>
    </CFOUTPUT>
</TABLE>
```

Допустим, если бы была необходимость вывести 5 записей, начиная с шестой позиции, для этого достаточно было бы добавить еще два атрибута:

```
<CFOUTPUT QUERY="GetTour" STARTROW="6" MAXROWS="5">
```

На рис. 11.5 представлен список всех круизов, зарегистрированных в базе данных, в рамках проекта "My-Line Cruises".

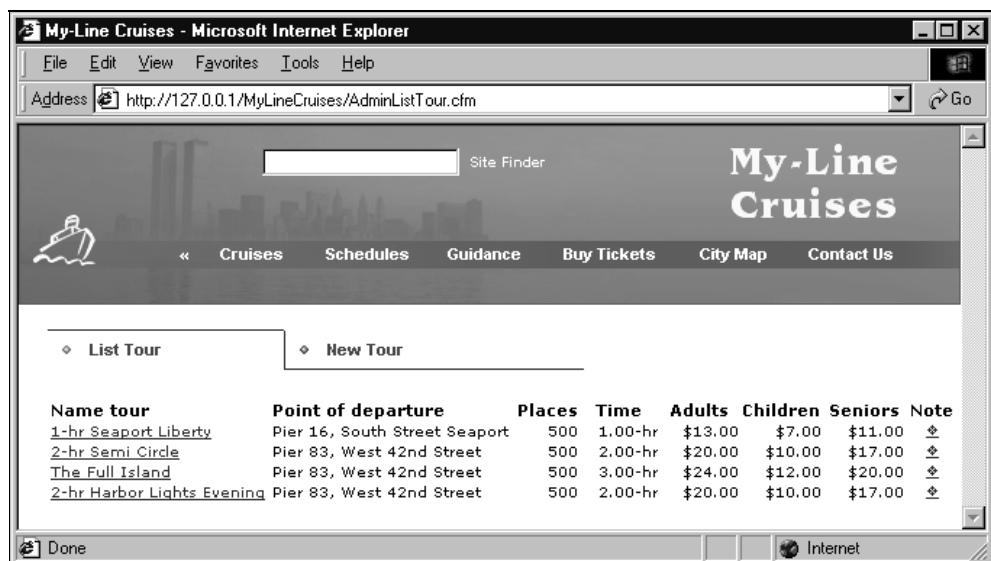


Рис. 11.5. Пример вывода списка круизов

Далее продемонстрируем полный код страницы AdminListTour.cfm, позволяющей решить описанную выше задачу (листинг 11.5).

Листинг 11.5. Код страницы AdminListTour.cfm

```
<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <style type="text/css">
  <!--
    #Tour { position:absolute; left:20; top:140; width:200; height:100;
      margin:0; z-index:0;}
  -->
</style>
<script language="JavaScript">
```

```

        if (ie) {document.write('<link rel="stylesheet" type="text/css"
                                href="main_ie.css" title="master">');}

</script>
</HEAD>
<BODY onload="init(1)"
        bgcolor="#CFOUTPUT#application.BGColorPage#</CFOUTPUT>"
        text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
    SELECT Key_tou, Name_tou, PointDepart_tou, NumLandingPlaces_tou,
           Time_tou, CostAdult_tou, CostChildren_tou, CostSenior_tou
    FROM Tour
</CFQUERY>
<DIV id="Tour">
    <TABLE width="370" border="0" cellpadding="0" cellspacing="0">
        <TR>
            <TR><TD colspan="2" bgcolor="#122c6d" height=1></TD>
                <TD colspan="2" height=1></TD></TR>
            <TD class="menleft">
                <IMG name="bull1" src="bullet2.gif" width="20" height="20"
                    border="0"></TD>
                <TD width="155" class="menleft">List Tour</TD>
                <TD bgcolor="#122c6d"></TD>
                <TD class="menleft"><a href="AdminNewTour.cfm"
                    onMouseOver="mouse_over_bull('bull2', 'Tour');
                    window.status='New Tour'; return true"
                    onMouseOut="mouse_out_bull('bull2', 'Tour');
                    window.status=''; return true">
                    <IMG name="bull2" src="bullet.gif" width="20" height="20"
                        border="0"></a></TD>
                <TD width="215"><a href="AdminNewTour.cfm"
                    onMouseOver="mouse_over_bull('bull2', 'Tour');"
                    onMouseOut="mouse_out_bull('bull2', 'Tour');"
                        class="menleft">
                        New Tour</a></td>
            <TR><TD colspan="2" height=1></TD>
                <TD colspan="4" bgcolor="#122c6d" height=1></TD></TR>
        </TR>
    </TABLE>

```

```

<br>
<TABLE width="630" border="0" cellspacing="2" cellpadding="0">
  <TR align="center">
    <TD class="txtTitleTab" width="160" align="left">Name tour</TD>
    <TD class="txtTitleTab" width="175" align="left">
      Point of departure</TD>
    <TD class="txtTitleTab" width="40">Places</td>
    <TD class="txtTitleTab" width="55">Time</TD>
    <TD class="txtTitleTab" width="55">Adults</TD>
    <TD class="txtTitleTab" width="55">Children</TD>
    <TD class="txtTitleTab" width="55">Seniors</TD>
    <TD class="txtTitleTab" width="35">Note</TD>
  </TR>
  <CFOUTPUT QUERY="GetTour">
    <TR align="right">
      <TD class="txtTab" align="left">
        <a href="AdminUpdateTour.cfm?key_tou=#Key_tou#">
          #Name_tou#</a></td>
      <TD class="txtTab" align="left">#PointDepart_tou#</TD>
      <TD class="txtTab">#NumLandingPlaces_tou#</TD>
      <TD class="txtTab">#Time_tou#-hr</TD>
      <TD class="txtTab">$#CostAdult_tou#</TD>
      <TD class="txtTab">$#CostChildren_tou#</TD>
      <TD class="txtTab">$#CostSenior_tou#</TD>
      <TD class="txtTab" align="center">
        <a href="AdminNoteTour.cfm?key_tou=#Key_tou#"
          target="_blank">
          <FONT face="Wingdings">±</FONT></a></TD>
    </TR>
  </CFOUTPUT>
</TABLE>
</DIV>
</BODY>
</HTML>

```

Обращаем ваше внимание на то, что в данном примере встречаются две ссылки: на страницу AdminUpdateTour.cfm, предназначенную для обновления выбранной записи, и страницу AdminNoteTour.cfm, служащую для вывода дополнительной информации по текущей записи. Что касается обновления данных, то об этом чуть позже.

Сейчас же рассмотрим страницу AdminNoteTour.cfm, куда, как видно из кода страницы AdminListTour.cfm, передается значение поля Key_tou текущей записи. Это позволяет определить условие запроса для вывода дополнительной информации, и т. к. значение поля Key_tou является уникальным по определению, то и результат выборки будет включать одну необходимую запись.

В листинге 11.6 приведен полный код страницы AdminNoteTour.cfm.

Листинг 11.6. Код страницы AdminNoteTour.cfm

```
<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <script language="JavaScript">
    if (ie) {document.write('<link rel="stylesheet" type="text/css"
      href="main_ie.css" title="master">');}
  </script>
</HEAD>
<BODY bgcolor="#application.BGColorPage#</CFOUTPUT>"
  text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
  <CFIF isDefined("URL.key_tou")>
    <CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
      SELECT Name_tou, Note_tou, Photo_tou
      FROM Tour WHERE Key_tou = #Url.key_tou#
    </CFQUERY>
    <CFOUTPUT QUERY="GetTour">
      <p class="title">#Name_tou#</p>
      <TABLE width="100%" border="0" cellspacing="2" cellpadding="0">
        <TR>
          <TD>
            <CFIF Photo_tou contains "gif">
              <IMG src="#Photo_tou#" alt="" border="0">
            </CFIF>
          </TD>
          <TD class="txt">#Note_tou#</TD>
        </TR>
      </TABLE>
    </CFOUTPUT>
```

```
<CFELSE>  
    <script language="JavaScript">self.close();</script>  
</CFIF>  
</BODY>  
</HTML>
```

На рис. 11.6 показана дополнительная информация по одному из выбранных круизов в рамках проекта "My-Line Cruises".

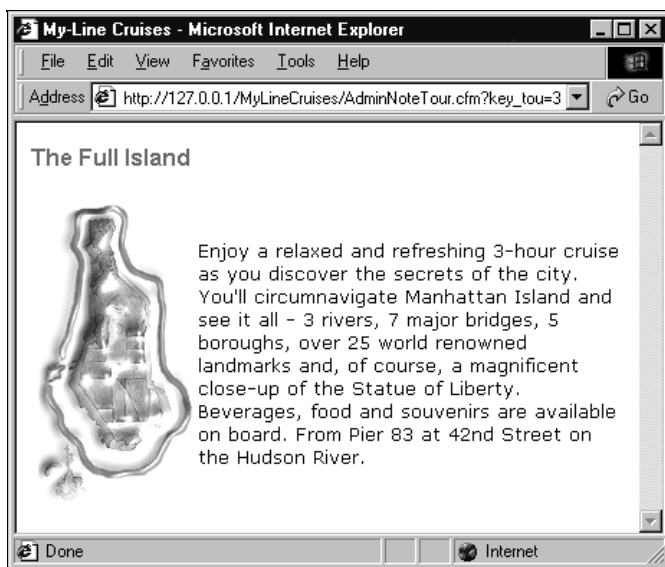


Рис. 11.6. Результат отображения страницы AdminNoteTour.cfm

Отметим, что в случае отсутствия переменной `URL.key_tou` страница AdminNoteTour.cfm автоматически закрывается.

Также вывести данные в ColdFusion можно с помощью тега `<CFTABLE>` в сочетании с тегом `<CFCOL>`. О чем, собственно говоря, и пойдет речь в следующем разделе с предварительным описанием тегов `<CFTABLE>` и `<CFCOL>`.

Тег `<CFTABLE>`

Синтаксис:

```
<CFTABLE QUERY = "query_name"  
    MAXROWS = "maxrows_table"  
    COLSPACING = "number_of_spaces"  
    HEADERLINES = "number_of_lines"  
    HTMLTABLE = "Yes | No"
```

```
BORDER = "Yes | No"
COLHEADERS = "Yes | No"
STARTROW = "row_number">
</CFTABLE>
```

В табл. 11.5 представлено описание атрибутов тега <CFTABLE>.

Таблица 11.5. Описание атрибутов тега <CFTABLE>

Атрибут	Описание
QUERY	Имя запроса. Обязательный атрибут. Для вывода результатов запроса внутри блока <CFTABLE>
MAXROWS	Максимальное количество записей, которое следует выводить внутри блока <CFTABLE>. Необязательный атрибут
COLSPACING	Пространство между колонками. Необязательный атрибут. По умолчанию равен 2
HEADERLINES	Высота между заголовком и первым рядом. По умолчанию значение равно 2, что соответствует одному интервалу
HTMLTABLE	Формирование страницы в формате HTML 3.0. Необязательный атрибут. Возможные значения: • Yes — да; • No — нет (по умолчанию)
BORDER	Рамка таблицы. Необязательный атрибут. Имеет смысл использовать при включении атрибута HTMLTABLE. Возможные значения: • Yes — да; • No — нет (по умолчанию)
COLHEADERS	Отображение заголовка таблицы. Необязательный атрибут. Возможные значения: • Yes — да; • No — нет (по умолчанию)
STARTROW	Номер записи, с которой следует выводить полученные данные. Необязательный атрибут

Тег <CFCOL>

Синтаксис:

```
<CFCOL HEADER = "column_header_text"
WIDTH = "number_indicating_width_of_column"
ALIGN = "Left | Right | Center"
TEXT = "text">
```

В табл. 11.6 представлено описание атрибутов тега `<CFCOL>`.

Таблица 11.6. Описание атрибутов тега `<CFCOL>`

Атрибут	Описание
HEADER	Текст заголовка колонки. Обязательный атрибут
WIDTH	Ширина колонки в символах. Необязательный атрибут. По умолчанию значение равно 20
ALIGN	Выравнивание текста внутри колонки. Необязательный атрибут. Возможные значения: • Left — по левому краю (по умолчанию); • Right — по правому краю; • Center — по центру
TEXT	Текст содержимого колонки. Текст может состоять из различных комбинаций и включать HTML-теги. Необязательный атрибут

Пример сочетания тегов `<CFTABLE>` и `<CFCOL>`

Мы не будем здесь приводить полный код страницы, покажем лишь, как можно было бы решить предыдущую задачу, где необходимо вывести все круизы, зарегистрированные в базе данных.

Напомним, что для решения данной задачи предварительно был построен запрос вот таким образом:

```
<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
    SELECT Key_tou, Name_tou, PointDepart_tou, NumLandingPlaces_tou,
           Time_tou, CostAdult_tou, CostChildren_tou, CostSenior_tou
    FROM Tour
</CFQUERY>
```

Далее выведем список полученных данных с использованием тегов `<CFTABLE>` и `<CFCOL>`:

```
<CFTABLE QUERY="GetTour" COLHEADERS="Yes" COLSPACING="0" HEADERLINES="1">
    <CFCOL HEADER = "<b>Name tour</b>" WIDTH = "27" ALIGN = "Left"
    TEXT = "<a href='AdminUpdateTour.cfm?key_tou=#Key_tou#'>#Name_tou#</a>"
    <CFCOL HEADER = "<b>Point of departure</b>" WIDTH = "28" ALIGN = "Left"
    TEXT = "#PointDepart_tou#"
    <CFCOL HEADER = "<b>Places</b>" WIDTH = "5" ALIGN = "Right"
    TEXT = "#NumLandingPlaces_tou#">
```



```

<CFCOL HEADER = "<b>Time</b>" WIDTH = "9" ALIGN = "Right"
    TEXT = "#Time_tou#-hr">
<CFCOL HEADER = "<b>Adults</b>" WIDTH = "9" ALIGN = "Right"
    TEXT = "$#CostAdult_tou#">
<CFCOL HEADER = "<b>Children</b>" WIDTH = "9" ALIGN = "Right"
    TEXT = "$#CostChildren_tou#">
<CFCOL HEADER = "<b>Seniors</b>" WIDTH = "9" ALIGN = "Right"
    TEXT = "$#CostSenior_tou#">
<CFCOL HEADER = "<b>Note</b>" WIDTH = "6" ALIGN = "Center"
    TEXT = "<a href='AdminNoteTour.cfm?key_tou=#Key_tou#'
        target='_blank'><FONT face='Wingdings'>±</FONT></a>">
</CFTABLE>

```

Модификация данных

Для модификации данных в ColdFusion можно воспользоваться несколькими вариантами: SQL-командой UPDATE внутри уже описанного нами тега <CFQUERY>, тегом <CFUPDATE>, а также тегом <CFGRID>, указываемого в рамках тега <CFFORM>. В данной главе будут рассмотрены только первые два способа.

Прежде чем привести пример по модификации данных, рассмотрим синтаксис SQL-команды UPDATE.

Команда UPDATE

Синтаксис:

```
UPDATE table_name1SET column_name1 = expression1 [, column_name2 = expression2 ...] WHERE filter_condition1 [AND | OR filter_condition2 ...]
```

Параметры:

❑ UPDATE table_name1

Задаёт таблицу, в которой необходимо изменить записи.

❑ SET column_name1 = expression1 [, column_name2 = expression2...]

Задаёт обновляемые столбцы и их новые значения. Если предложение WHERE опущено, каждая строка одного столбца обновляется одним и тем же значением.

❑ WHERE filter_condition1 [AND | OR filter_condition2 ...]

Определяет, какие записи следует изменять.

Условие filter_condition задаёт критерий, которому должны удовлетворять изменяемые записи. Можно включить сколько угодно условий фильт-

рования, объединяя их операторами AND и OR. Можно также использовать оператор NOT, инвертирующий значение логического выражения.

Замечание

Команда UPDATE может модифицировать записи только в одной таблице.

Пример команды UPDATE

Предположим, нам необходимо организовать возможность модификации данных для уже зарегистрированных покупателей.

В этом случае требуется создать форму для редактирования данных с последующим изменением этих данных для конкретного покупателя. Тогда в нашем случае запрос в ColdFusion будет выглядеть следующим образом:

```
<CFQUERY NAME="UpdateTraveler" DATASOURCE="#application.dsn#">
UPDATE Traveler
    SET EMail_tra = '#Form.Email#',
        FirstName_tra = '#Form.FirstName#',
        LastName_tra = '#Form.LastName#',
        MiddleInitial_tra = '#Form.MiddleInitial#',
        Password_tra = '#Form.Password#',
        PhoneNum_tra = '#Form.PhoneNum#'
    WHERE Key_tra = #Session.traveler.id#
</CFQUERY>
```

Здесь в предложении SET перечислены обновляемые столбцы и их новые значения.

Следующим этапом обновляем переменную сессии Session.traveler.name, поскольку значение этой переменной есть ни что иное, как сочетание значений двух модифицируемых полей, которые могли быть изменены. После чего открываем основную страницу и с помощью тега <CFEXIT> прекращаем выполнение последующего кода текущей страницы. Этот код таков:

```
<CFSET Session.traveler.name = #Form.FirstName#&" "&#Form.LastName#>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=BuyTickets.cfm">
<CFEXIT>
```

На рис. 11.7 показана форма для модификации данных о покупателе в текущей сессии в рамках проекта "My-Line Cruises".

В листинге 11.7 представлен полный код страницы UpdateTra.cfm, позволяющей решить поставленную перед нами задачу.

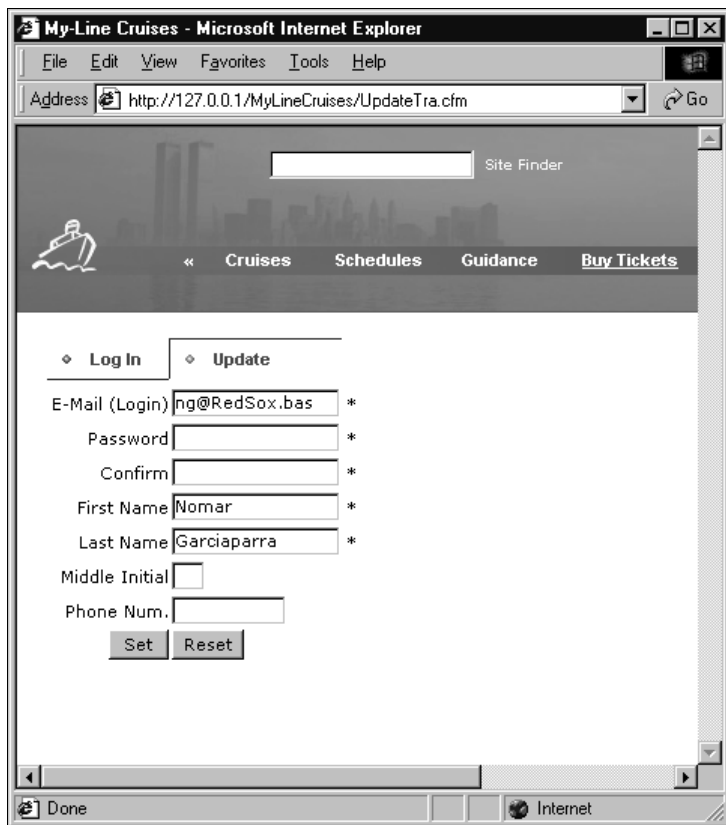


Рис. 11.7. Пример формы редактирования сведений о покупателе

Листинг 11.7. Код страницы UpdateTra.cfm

```
<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <style type="text/css">
  <!--
  #Update { position:absolute; left:20; top:140; width:200; height:100;
    margin:0;z-index:0;}
  -->
```

```
</style>
<script language="JavaScript">
    if (ie) {document.write('<link rel="stylesheet" type="text/css"
        href="main_ie.css" title="master">');}
</script>
</HEAD>
<BODY onload="init(1)"
    bgcolor="<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
    text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<CFIF isDefined("Session.traveler.id")>
    <CFIF isDefined("Form.Email")>
        <CFSET Disuccessful = 1>
        <CFSET msgError = "Warning Error!">
        <CFIF Len(Form.Email) le 5 and Form.Email does not contain "@">
            <CFSET msgError = msgError & "<br>" & #application.msgError1#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Len(Form.Password) le 4>
            <CFSET msgError = msgError & "<br>" & #application.msgError2#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF IsNumeric(Form.Password)>
            <CFSET msgError = msgError & "<br>" & #application.msgError3#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Form.Password not equal Form.Confirm>
            <CFSET msgError = msgError & "<br>" & #application.msgError4#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Len(Form.FirstName) le 0>
            <CFSET msgError = msgError & "<br>" & #application.msgError5#>
            <CFSET Disuccessful = 0>
        </CFIF>
        <CFIF Len(Form.LastName) le 0>
            <CFSET msgError = msgError & "<br>" & #application.msgError6#>
            <CFSET Disuccessful = 0>
        </CFIF>
```

```

<CFIF YesNoFormat(Disuccessful) equal "No"> <!--- Ошибки (Errors) --->
    <CFSET email = #Form.EMail#>
    <CFSET fname = #Form.FirstName#>
    <CFSET lname = #Form.LastName#>
    <CFSET middle = #Form.MiddleInitial#>
    <CFSET phonen = #Form.PhoneNum#>
<CFELSE>
    <!--- Обновление записи (Update record) --->
    <CFQUERY NAME="UpdateTraveler" DATASOURCE="#application.dsn#">
        UPDATE Traveler
        SET EMail_tra = '#Form.EMail#',
            FirstName_tra = '#Form.FirstName#',
            LastName_tra = '#Form.LastName#',
            MiddleInitial_tra = '#Form.MiddleInitial#',
            Password_tra = '#Form.Password#',
            PhoneNum_tra = '#Form.PhoneNum#'
        WHERE Key_tra = #Session.traveler.id#
    </CFQUERY>
    <CFSET Session.traveler.name = #Form.FirstName# & " " &
        #Form.LastName#>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=BuyTickets.cfm">
    <CFEXIT>
</CFIF>
<CFELSE>
    <CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
        SELECT *
        FROM Traveler WHERE Key_tra = #Session.traveler.id#
    </CFQUERY>
    <CFOUTPUT QUERY="GetTraveler">
        <CFSET email = #EMail_tra#>
        <CFSET fname = #FirstName_tra#>
        <CFSET lname = #LastName_tra#>
        <CFSET middle = #MiddleInitial_tra#>
        <CFSET phonen = #PhoneNum_tra#>
        <CFSET msgError = "">
    </CFOUTPUT>
</CFIF>
<DIV id="Update">

```

```

<FORM action='UpdateTra.cfm' method='post' name='form1'>
  <TABLE border='0' cellspacing='0' cellpadding='0'>
    <TR>
      <TR><TD colspan="2" height=1</TD>
        <TD colspan="4" bgcolor="#122c6d" height=1</TD></TR>
      <TD class="menleft"><a href="BuyTickets.cfm"
        onMouseOver="mouse_over_bull('bull12', 'Update');
        window.status='Sign Up'; return true"
        onMouseOut="mouse_out_bull('bull12', 'Update');
        window.status=''; return true">
          </a></td>
      <TD width="55"><a href="BuyTickets.cfm"
        onMouseOver="mouse_over_bull('bull12', 'Update');"
        onMouseOut="mouse_out_bull('bull12', 'Update');"
        class="menleft">
          Log In</a></TD>
      <TD bgcolor="#122c6d"></TD>
      <TD class="menleft">
        </TD>
      <TD width="82" class="menleft">Update</TD>
    <TR><td colspan="2" bgcolor="#122c6d" height=1</TD>
      <TD colspan="2" height=1</TD></TR>
  </TR>
</TABLE>
<table width="300" border="0" cellspacing="2" cellpadding="0">
  <TR><TD colspan="2" align="center" class="txtError">
    <CFOUTPUT>#msgError#</CFOUTPUT></TD></TR>
  <TR><TD align="right" class="txtForm1">
    E-Mail (Login)</td><td width="215" class="txtForm1">
      <input name='EMail' size='16'
        value="<CFOUTPUT>#email#</CFOUTPUT>">&nbsp;  *</td></TR>
  <TR><TD align="right" class="txtForm1">
    Password</td><td class="txtForm1">
      <input type='password' name='Password'
        size='16'>&nbsp;  *</td></TR>
  <TR><TD align="right" class="txtForm1">

```

```

Confirm</td><td class="txtForm1">
  <input type='password' name='Confirm'
    size='16'>&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
  First Name</td><td class="txtForm1">
    <input name='FirstName' size='16'
      value="<CFOUTPUT>#fname#</CFOUTPUT>">&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
  Last Name</td><td class="txtForm1">
    <input name='LastName' size='16'
      value="<CFOUTPUT>#lname#</CFOUTPUT>">&nbsp;  *</TD></TR>
<TR><TD align="right" class="txtForm1">
  Middle Initial</td><td>
    <input name='MiddleInitial' size='1'
      value="<CFOUTPUT>#middle#</CFOUTPUT>"></TD></TR>
<TR><TD align="right" class="txtForm1">
  Phone Num.</td><td>
    <input name='PhoneNum' size='10'
      value="<CFOUTPUT>#phonen#</CFOUTPUT>"></TD></TR>
<TR><td align="right">
  <input type='submit' value=' Set '></TD><TD>
  <input type='reset' value='Reset'></TD></TR>
</TABLE>

```

```
</FORM>
```

```
</DIV>
```

```
<CFELSE>
```

```
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=BuyTickets.cfm">
```

```
</CFIF>
```

```
</BODY>
```

```
</HTML>
```

Отметим, что сообщения об ошибках определены в конфигурационном файле `Application.cfm` так, как это показано в *разд. "Пример файла Application.cfm" гл. 10*.

Также для модификации данных в ColdFusion можно использовать тег `<CFUPDATE>`, о чем и пойдет речь в следующем разделе с предварительным описанием этого тега.

Тег <CFUPDATE>

Синтаксис:

```
<CFUPDATE DATASOURCE = "ds_name"
          DBTYPE = "type"
          DBSERVER = "dbms"
          DBNAME = "database name"
          TABLENAME = "tbl_name"
          CONNECTSTRING = "connection string"
          TABLEOWNER = "owner"
          TABLEQUALIFIER = "tbl_qualifier"
          USERNAME = "username"
          PASSWORD = "password"
          PROVIDER = "COMProvider"
          PROVIDERDSN = "datasource"
          FORMFIELDS = "formfield1, formfield2, ...">
```

Атрибуты тега <CFUPDATE> полностью идентичны с атрибутами уже описанного тега <CFINSERT> (см. табл. 11.3).

Пример тега <CFUPDATE>

Чтобы продемонстрировать использование тега <CFUPDATE>, предназначенного для модификации данных, рассмотрим пример, позволяющий изменять информацию об уже зарегистрированных экскурсиях и круизах.

Для решения данной задачи необходимо создать форму, предназначенную для редактирования необходимых данных с целью их последующего изменения по выбранному круизу. Тогда для нашего примера код модификации данных будет выглядеть следующим образом:

```
<CFUPDATE DATASOURCE = "#application.dsn#"
          TABLENAME = "Tour"
          FORMFIELDS = "Key_tou,Name_tou,PointDepart_tou,
                        NumLandingPlaces_tou,Time_tou,CostAdult_tou,
                        CostChildren_tou,CostSenior_tou,Photo_tou,Note_tou">
```

На рис. 11.8 показана форма для редактирования данных по выбранному круизу в рамках проекта "My-Line Cruises".

В листинге 11.8 приведен полный код страницы AdminUpdateTour.cfm, позволяющей решить описанную выше задачу.

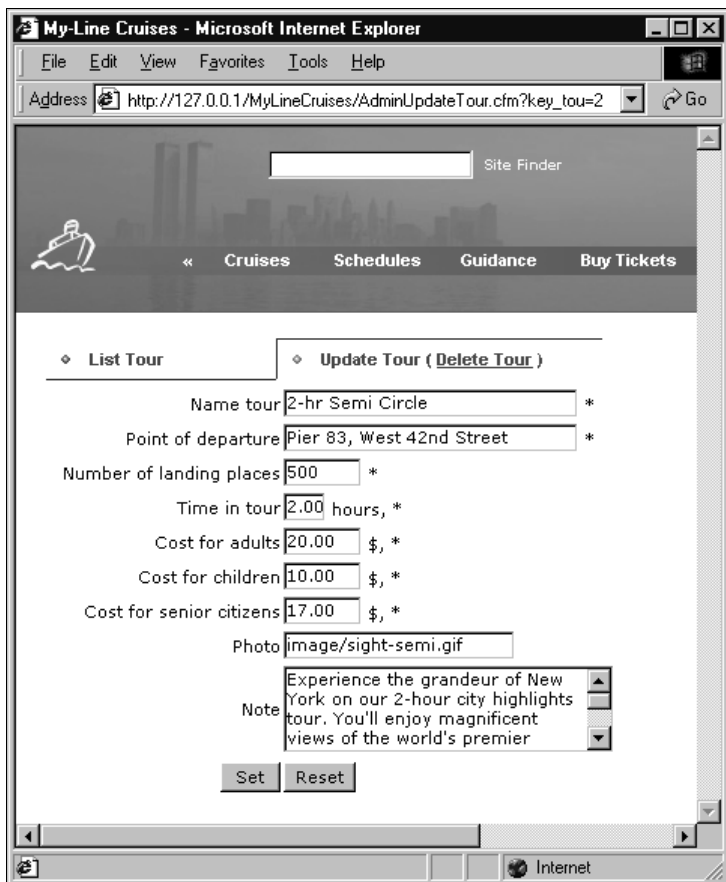


Рис. 11.8. Пример формы редактирования данных о круизе

Листинг 11.8. Код страницы AdminUpdateTour.cfm

```
<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <style type="text/css">
  <!--
  #Tour { position:absolute; left:20; top:140; width:200; height:100;
    margin:0; z-index:0;}
```

```

-->
</style>
<script language="JavaScript">
    if (ie) {document.write('<link rel="stylesheet" type="text/css"
        href="main_ie.css" title="master">');}
</script>
</HEAD>
<BODY onload="init(1)"
    bgcolor="<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
    text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<CFIF isDefined("key_tou")>
    <CFIF isDefined("Form.Name_tou")>
        <CFSET Disuccessful = 1>
        <CFSET msgError = "Warning Error!">
        <CFIF Len(Form.Name_tou) le 0 or Len(Form.PointDepart_tou) le 0 or
            Len(Form.NumLandingPlaces_tou) le 0 or Len(Form.Time_tou) le 0 or
            Len(Form.CostAdult_tou) le 0 or Len(Form.CostChildren_tou) le 0 or
            Len(Form.CostSenior_tou) le 0>
            <CFSET msgError = msgError & "<br>" & "The fields marked with
                a symbol *, should not be empty!">
            <CFSET Disuccessful = 0>
    </CFIF>
<CFIF YesNoFormat(Disuccessful) equal "No"> <!-- Errors --->
    <CFSET Name_tou = #Form.Name_tou#>
    <CFSET PointDepart_tou = #Form.PointDepart_tou#>
    <CFSET NumLandingPlaces_tou = #Form.NumLandingPlaces_tou#>
    <CFSET Time_tou = #Form.Time_tou#>
    <CFSET CostAdult_tou = #Form.CostAdult_tou#>
    <CFSET CostChildren_tou = #Form.CostChildren_tou#>
    <CFSET CostSenior_tou = #Form.CostSenior_tou#>
    <CFSET Photo_tou = #Form.Photo_tou#>
    <CFSET Note_tou = #Form.Note_tou#>
<CFELSE>
    <!-- Обновление записи (Update record) --->
    <CFUPDATE DATASOURCE = "#application.dsn#"
        TABLENAME = "Tour"
        FORMFIELDS = "Name_tou,PointDepart_tou,

```

```

        NumLandingPlaces_tou,
        Time_tou, CostAdult_tou, CostChildren_tou,
        CostSenior_tou, Photo_tou, Note_tou">
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=AdminListTour.cfm">
    <CFEXIT>
</CFIF>
<CFELSE>
    <CFSET msgError = "">
    <CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
        SELECT *
        FROM Tour WHERE Key_tou = '#key_tou#'
    </CFQUERY>
    <CFOUTPUT QUERY="GetTour">
        <CFSET Name_tou = #Name_tou#>
        <CFSET PointDepart_tou = #PointDepart_tou#>
        <CFSET NumLandingPlaces_tou = #NumLandingPlaces_tou#>
        <CFSET Time_tou = #Time_tou#>
        <CFSET CostAdult_tou = #CostAdult_tou#>
        <CFSET CostChildren_tou = #CostChildren_tou#>
        <CFSET CostSenior_tou = #CostSenior_tou#>
        <CFSET Photo_tou = #Photo_tou#>
        <CFSET Note_tou = #Note_tou#>
    </CFOUTPUT>
</CFIF>
<DIV id="Tour">
    <FORM action='AdminUpdateTour.cfm' method='post' name='form1'>
        <TABLE width="370" border="0" cellspacing="0" cellpadding="0">
            <TR>
                <TR><TD colspan="2" height=1></TD>
                <TD colspan="4" bgcolor="#122c6d" height=1></TD></TR>
                <TD class="menleft"><a href="AdminListTour.cfm"
                    onMouseOver="mouse_over_bull('bull12', 'Tour');
                    window.status='List Tour'; return true"
                    onMouseOut="mouse_out_bull('bull12', 'Tour');
                    window.status=''; return true">
                    <IMG name="bull12" src="bullet.gif" width="20" height="20"
                        border="0"></a></TD>
                <TD width="155"><a href="AdminListTour.cfm"
                    onMouseOver="mouse_over_bull('bull12', 'Tour');"
```

```

        onMouseOut="mouse_out_bull('bull12', 'Tour');
        "class="menleft">

        List Tour</a></TD>
<TD bgcolor="#122c6d"></TD>
<TD class="menleft">
    <IMG name="bull11" src="bullet2.gif" width="20" height="20"
        border="0"></td>
<TD width="215" class="menleft">
    Update Tour    <a
href="javascript:del('AdminDeleteTour.cfm?key_tou=<CFOUTPUT>#key_tou#
</CFOUTPUT>');" class="menleft_u">Delete Tour</a>
</TD>
<TR><td colspan="2" bgcolor="#122c6d" height=1></TD>
<TD colspan="2" height=1></TD></TR>
</TR>
</TABLE>
<TABLE width="390" border="0" cellpadding="2" cellspacing="0">
<TR><td colspan="2" align="center" class="txtError">
    <CFOUTPUT>#msgError#</CFOUTPUT></TD></TR>
<TR><td align="right" class="txtForm1">
    Name tour</td><td width="230" class="txtForm1">
    <input name='Name_tou' size='30'
        value="<CFOUTPUT>#Name_tou#</CFOUTPUT>">&nbsp;   *</td></TR>
<TR><td align="right" class="txtForm1">
    Point of departure</td><td class="txtForm1">
    <input name='PointDepart_tou' size='30'
        value="<CFOUTPUT>#PointDepart_tou#</CFOUTPUT>">&nbsp;   *</td></TR>
<TR><td align="right" class="txtForm1">
    Number of landing places</td><td class="txtForm1">
    <input name='NumLandingPlaces_tou' size='6'
        value="<CFOUTPUT>#NumLandingPlaces_tou#</CFOUTPUT>">&nbsp;   *</td></TR>
<TR><td align="right" class="txtForm1">
    Time in tour</td><td class="txtForm1">
    <input name='Time_tou' size='2'
        value="<CFOUTPUT>#Time_tou#</CFOUTPUT>">&nbsp;   hours,&nbsp;   *</td></TR>

```

```

<TR><TD align="right" class="txtForm1">
    Cost for adults</td><td class="txtForm1">
        <input name='CostAdult_tou' size='6'
            value="<CFOUTPUT>#CostAdult_tou#</CFOUTPUT>">&nbsp;&nbsp;&nbsp;$,&nbsp;&nbsp;&nbsp;*
    </TD></TR>

<TR><TD align="right" class="txtForm1">
    Cost for children</td><td class="txtForm1">
        <input name='CostChildren_tou' size='6'
            value="<CFOUTPUT>#CostChildren_tou#</CFOUTPUT>">&nbsp;&nbsp;&nbsp;$,&nbsp;&nbsp;&nbsp;*
    </TD></TR>

<TR><TD align="right" class="txtForm1">
    Cost for senior citizens</td><td class="txtForm1">
        <input name='CostSenior_tou' size='6'
            value="<CFOUTPUT>#CostSenior_tou#</CFOUTPUT>">&nbsp;&nbsp;&nbsp;$,&nbsp;&nbsp;&nbsp;*
    </TD></TR>

<TR><TD align="right" class="txtForm1">
    Photo</TD><TD>
        <input name='Photo_tou' size='23'
            value="<CFOUTPUT>#Photo_tou#</CFOUTPUT>"></TD></TR>

<TR><TD align="right" class="txtForm1">
    Note</TD><TD>
        <textarea name = "Note_tou" cols = "33"
            rows = "4"><CFOUTPUT>#Note_tou#</CFOUTPUT>
        </textarea>
    </TD></TR>

<TR><TD><input type="Hidden" name="key_tou"
            value="<CFOUTPUT>#key_tou#</CFOUTPUT>"></td><td>

<TR><TD align="right">
    <input type='submit' value=' Set '></TD><TD>
    <input type='reset' value='Reset'></TD></TR>

</TABLE>

</FORM>

</DIV>

<CFELSE>

    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=AdminListTour.cfm">

</CFIF>

</BODY>

</HTML>

```

Обращаем ваше внимание, что на страницу AdminUpdateTour.cfm пользователь попадает со страницы AdminListTour.cfm, откуда и передается переменная `key_tou`, с помощью которой производится выборка необходимых данных.

В рассмотренном примере встречается ссылка на страницу AdminDeleteTour.cfm, предназначенную для удаления текущей записи, о чем, собственно говоря, и пойдет речь далее.

Удаление данных

Для удаления данных в ColdFusion существует опять же несколько способов: применение SQL-команды `DELETE` внутри уже описанного нами тега `<CFQUERY>`, указание тега `<CFGRID>`, используемого в рамках тега `<CFFORM>`. Впрочем, тег `<CFGRID>` позволяет только визуализировать процесс удаления данных, а для непосредственного удаления, так или иначе, необходимо использовать команду `DELETE`. В данной главе будет рассмотрен только первый способ.

Команда **DELETE**

Синтаксис:

```
DELETE FROM table_name
    [WHERE filter_condition1 [AND | OR filter_condition2 ...]]
```

Параметры:

❑ `FROM table_name`

Задает таблицу, в которой предполагается удаление записей.

❑ `WHERE filter_condition1 [AND | OR filter_condition2 ...]`

Указывает на удаление только определенных записей.

Условие `filter_condition` устанавливает критерий, которому должны удовлетворять записи, предназначенные для удаления. Можно задать сколько угодно условий фильтрования, объединяя их с помощью операторов `AND` и `OR`. Можно также пользоваться оператором `NOT`, инвертирующим значение логического выражения.

Пример команды **DELETE**

Предположим, нам необходимо в режиме обновления данных по тому или иному круизу предоставить возможность удалить текущую запись.

В этом случае требуется лишь установить ссылку на страницу, отвечающую за удаление данных с обязательным включением в URL-адрес значения уникального ключа для удаления только текущей записи. К примеру, так:

```
<a href="AdminDeleteTour.cfm?key_tou=
    <CFOUTPUT>#key_tou#</CFOUTPUT>'" ">Delete Tour</a>
```

А уже непосредственно запрос на удаление в указанной странице может выглядеть следующим образом:

```
<CFQUERY NAME="DeleteTour" DATASOURCE="#application.dsn#">
    DELETE Tour WHERE Key_tou = '#key_tou#'
</CFQUERY>
```

Однако правильно было бы перед удалением записи обращаться к пользователю за подтверждением удаления, и здесь могут быть разные решения. Например, можно предварительно выводить модальное окно, подобное тому, что представлено на рис. 11.9.

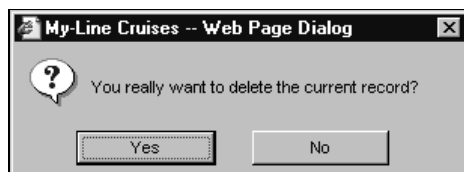


Рис. 11.9. Пример модального окна для подтверждения удаления текущей записи

Для реализации подтверждения на удаление мы несколько преобразуем ссылку:

```
<a href="javascript:del('AdminDeleteTour.cfm?key_tou=
    <CFOUTPUT>#key_tou#</CFOUTPUT>');" class="menleft_u">Delete Tour</a>
```

где `del` — заранее созданная функция и помещенная в основной файл JavaScript `main.js`. Приведем код этой функции:

```
function del(url) {
    if(ie){
        _del = self.showModalDialog ("DialogQuestion.cfm?text_=You really
                                     want to delete the current record?",
                                     null,"status:no;center:yes;help:no;
                                     minimize:no;maximize:no;border:thin;
                                     statusbar:no;dialogWidth:305px;
                                     dialogHeight:110px");

        if(_del==true) {
            location.href=url;
        }
    } else {
        location.href=url;
    }
}
```

Необходимо отметить, что с помощью данной функции мы открываем модальное окно только для браузера MS Internet Explorer. Для остальных же браузеров запись удаляется без подтверждения. Впрочем, у вас есть возможность исправить данный недочет и самостоятельно решить эту задачу. Мы же приведем пример страницы DialogQuestion.cfm, исполняющей роль диалога подтверждения удаления записи.

Листинг 11.9. Код страницы DialogQuestion.cfm

```
<HTML>  
    <HEAD>  
        <CFINCLUDE TEMPLATE = "MyHEAD.cfm">  
        <link rel="STYLESHEET" type="text/css" href="main.css">  
        <style type="text/css">  
            <!--  
                #pic_ {position: absolute; left:10; top:10; margin:0;}  
            -->  
        </style>  
    </HEAD>  
  
    <BODY bgcolor="#C0C0C0">  
        <div id="pic_">  
              
        </div>  
        <br>  
        <p class="dialog_">  
            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
            ~~~~~  
        <CFIF isDefined("URL.text_")>  
            <CFOUTPUT>#URL.text_#</CFOUTPUT>  
        </CFIF>  
        </p>  
        <p ALIGN="center">  
            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
            <BUTTON ID="btnDelYes" style="height:23;">  
            <SPAN class="dialog">&nbsp;&nbsp;&nbsp;&nbsp;&~  
            Yes&nbsp;&nbsp;&~  
            ~~~~~  
            <BUTTON ID="btnDelNo" style="height:23;">
```


В заключение описания удаления данных посредством ColdFusion приведем пример страницы AdminDeleteTour.cfm, где, в общем-то, и совершается "обряд" удаления выбранного круиза.

```
<HTML>

<HEAD>

<CFINCLUDE TEMPLATE = "MyHEAD.cfm">

</HEAD>

<BODY bgcolor="<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
      text="Black" alink="#107098" link="#0C1C47" vlink="#003366">

<CFIF isDefined("key_tou")>

      <CFQUERY NAME="DeleteTour" DATASOURCE="#application.dsn#">

          DELETE Tour WHERE Key_tou = '#key_tou#'

      </CFQUERY>

</CFIF>

<META HTTP-EQUIV="Refresh" CONTENT="0;URL=AdminListTour.cfm">

</BODY>

</HTML>
```

Отметим лишь, что по мере удаления записей пользователь возвращается к списку еще пока существующих круизов.

Транзакции

Транзакция — это логическая единица работы, т. е. транзакция в приложении обеспечивает защиту путем выделения секции программного кода в качестве единицы восстановления.

Рассмотрим пример. Предположим, что таблица `TOUR` включает дополнительный атрибут `TotalTraveler`, представляющий собой общее число путешественников для каждого круиза. Значение `TotalTraveler` для любого круиза должно быть равным сумме купленных билетов. Покупка же билетов регистрируется в таблице `TICKET`. Данный пример предполагает, что речь идет о выполнении двух операций: добавлении новой записи в таблицу `TICKET` и модификации значения поля `TotalTraveler` в таблице `TOUR` для выбранного круиза. При этом в базе данных между двумя указанными операциями временно нарушается требование, что значение `TotalTraveler` для круиза равно сумме купленных билетов на конкретный круиз. Таким образом, логическая единица работы — не просто одиночная операция системы баз данных, а скорее согласование нескольких таких операций. И не допустимо, чтобы одна из операций была выполнена, а другая нет, поскольку база данных остается в несогласованном состоянии. В идеальном случае должны быть выполнены обе операции. Однако нельзя обеспечить абсолютную гарантию, что так и случится. Не исключена вероятность, что система может дать сбой между двумя операциями, или же на второй операции произойдет арифметическое переполнение и т. д. Система же, поддерживающая процесс транзакции, гарантирует, что если во время выполнения неких операций произошла ошибка, то все результаты операций будут аннулированы. Таким образом, транзакция или выполняется целиком, или полностью отменяется.

Для установки транзакции в ColdFusion необходимо использовать тег `<CFTRANSACTION>`.

Тег `<CFTRANSACTION>`

Синтаксис:

```
<CFTRANSACTION ACTION = "Begin | Commit | Rollback"
                  ISOLATION = "Read_Uncommitted | Read_Committed |
                              Repeatable_Read | Serializable">
</CFTRANSACTION>
```

В табл. 11.7 представлено описание атрибутов тега `<CFTRANSACTION>`.

Таблица 11.7. Описание атрибутов тега <CFTRANSACTION>

Атрибут	Описание
ACTION	Действие. Необязательный атрибут. Возможные значения: • <code>Begin</code> — начать транзакцию (по умолчанию); • <code>Commit</code> — применить изменения, совершенные с момента начала транзакции; • <code>Rollback</code> — отменить все изменения, сделанные с момента начала транзакции
ISOLATION	Уровень изоляции. Необязательный атрибут. Возможные значения: • <code>Read_Uncommitted</code> — незавершенное считывание, предназначено для "неаккуратных" считываний; • <code>Read_Committed</code> — завершенное считывание; • <code>Repeatable_Read</code> — повторяемое считывание; • <code>Serializable</code> — способность к упорядочению, по сути, является единственным безопасным уровнем

В пределах блока транзакции можно применять изменения, совершенные с момента начала транзакции, указав

```
<CFTRANSACTION ACTION = "commit"/>
```

либо отменить все изменения, добавив

```
<CFTRANSACTION ACTION = "rollback"/>
```

Пример тега <CFTRANSACTION>

Усложним задачу удаления круизов. В следующем примере (листинг 11.11) с использованием транзакции удаляется запись со сведениями о выбранном круизе и все связанные записи с данными о расписании и забронированных билетах.

Листинг 11.11. Код страницы AdminDeleteTour.cfm с добавлением блока транзакции

```
<HTML>
<HEAD>
    <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
</HEAD>
<BODY bgcolor="<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
    text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
```

```
<CFIF isDefined("key_tou")>
  <CFTRANSACTION ACTION="Begin">
    <CFQUERY NAME="DeleteTicket" DATASOURCE="#application.dsn#">
      DELETE Ticket
      WHERE Key_tou = '#key_tou#'
    </CFQUERY>
    <CFQUERY NAME="DeleteSchedule" DATASOURCE="#application.dsn#">
      DELETE Schedule_
      WHERE Key_tou = '#key_tou#'
    </CFQUERY>
    <CFQUERY NAME="DeleteTour" DATASOURCE="#application.dsn#">
      DELETE Tour
      WHERE Key_tou = '#key_tou#'
    </CFQUERY>
  </CFTRANSACTION ACTION="Commit"/>
</CFTRANSACTION>
</CFIF>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=AdminListTour.cfm">
</BODY>
</HTML>
```

Если при выполнении одной из команд `DELETE` возникает ошибка, транзакция выполнит откат, что позволит избежать конфликтов.

Резюме

В заключение следует отметить, что SQL-запросы являются одним из главных инструментов работы с данными, и если вам пока трудно дается написание SQL-запросов вручную, вы можете прибегнуть к помощи конструктора запросов ColdFusion Studio, подробно описанного в *разд. "Базы данных" гл. 4*.

Подведем итог. В данной главе мы познакомились и использовали CFML-теги: `<CFQUERY>`, `<CFINSERT>`, `<CFTABLE>`, `<CFCOL>`, `<CFUPDATE>`, `<CFTRANSACTION>`.



Построение форм

В предыдущих главах мы уже неоднократно использовали HTML-тег `<FORM>` позволяющий создавать формы для обработки данных. Впрочем, в ColdFusion существует тег `<CFFORM>`, предназначенный для той же цели, но при этом включающий в себя помимо текстовых полей ввода, флажков, переключателей (или радиокнопок) и списков, такие элементы, как дерево (tree), движок (slider), сетку данных (grid), а также предоставляющий возможность дополнять форму собственными Java-апплетами.

Помимо дополнительных элементов, позволяющих с легкостью обогатить собственные формы, применение тега `<CFFORM>` в сочетании с тегами ввода помогает контролировать поля ввода с помощью встроенного атрибута `VALIDATE`, тем самым облегчая написание кода. В то же время вы можете использовать HTML-тег `<FORM>` в комбинации с тегом `<CFFORM>`, используя атрибут `PASSTHROUGH`, но обо всем по порядку.

Тег формы `<CFFORM>` имеет следующие базовые управляющие элементы:

- ❑ `<CFINPUT>` — предназначен для размещения текстовых полей ввода (text input boxes), флажков (checkboxes) и переключателей (radio buttons);
- ❑ `<CFSELECT>` — задает список, позволяющий выбрать один или несколько вариантов.

Тег формы `<CFFORM>` может содержать управляющие элементы, основанные на Java-апплетах:

- ❑ `<CFTEXTINPUT>` — текстовое поле ввода с возможностью форматирования текста;
- ❑ `<CFSLIDER>` — движок, перемещаемый по вертикали или горизонтали, позволяющий визуально управлять числовыми величинами в указанном диапазоне;
- ❑ `<CFTREE>` — дерево, представляющее собой элемент управления для создания иерархического списка;
- ❑ `<CFGRID>` — сетку данных, позволяющую организовать работу с данными в виде сетки-таблицы со строками и колонками;
- ❑ `<CFAPPLET>` — собственные Java-апплеты.

Тег <CFFORM>

Синтаксис:

```
<CFFORM NAME = "name"
    ACTION = "form_action"
    PRESERVEDATA = "Yes | No"
    ENABLECAB = "Yes | No"
    ONSUBMIT = "javascript"
    TARGET = "window_name"
    ENCTYPE = "type"
    PASSTHROUGH = "HTML_attributes"
    CODEBASE = "URL"
    ARCHIVE = "URL">
```

```
    Элементы формы
</CFFORM>
```

В табл. 12.1 представлено описание атрибутов тега <CFFORM>.

Таблица 12.1. Описание атрибутов тега <CFFORM>

Атрибут	Описание
NAME	Наименование формы. Необязательный атрибут
ACTION	Имя ColdFusion-страницы, которая будет обрабатывать переменные формы. Обязательный атрибут
PRESERVEDATA	Определяет, отражать ли данные, которые были введены в форме, образованной тегом <CFFORM>, на странице, указанной атрибутом ACTION. Этот атрибут предназначен для управляющих элементов <CFSLIDER> и <CFTEXTINPUT>. Для управляющего элемента <CFTREE> позволяет раскрывать предварительно выбранные пункты. Для <CFTREE> атрибут COMPLETEPATH должен быть равен Yes. Возможные значения: • Yes — да; • No — нет (по умолчанию) Атрибут может использоваться, если страница с формой и страница, указанная атрибутом ACTION, являются одним и тем же файлом, либо если на странице, указанной атрибутом ACTION, установлены идентичные управляющие элементы с теми же именами. Необязательный атрибут
ENABLECAB	В версии ColdFusion Server 5.0 не используется

Таблица 12.1 (окончание)

Атрибут	Описание
ONSUBMIT	Определяет JavaScript-обработчик активизации формы. Применяется для проверки с помощью кода JavaScript правильности заполнения элементов формы. Необязательный атрибут
TARGET	Имя окна или фрейма, куда будет передан результат заполнения формы. Необязательный атрибут
ENCTYPE	MIME-тип для указания формата посылки данных. По умолчанию принимает значение <code>application/x-www-form-urlencoded</code> , рекомендуется не изменять. Атрибут включен для совместимости с предыдущими версиями. Необязательный атрибут
PASSTHROUGH	Установка возможности использования HTML-тега <code><FORM></code> в комбинации с тегом <code><CFFORM></code> для тега <code><CFINPUT></code> . Возможные значения: • Yes — да; • No — нет (по умолчанию). Необязательный атрибут
CODEBASE	URL для загрузки (download) дополнения (plug-ins) JRE. Только для Internet Explorer. Значение по умолчанию <code>/CFIDE/classes/cf-j2re-win.cab</code> . Необязательный атрибут
ARCHIVE	URL для загрузки (download) Java-классов управляющих средств (controls) ColdFusion. Значение по умолчанию <code>/CFIDE/classes/CFJava2.jar</code> . Необязательный атрибут

Тег `<CFINPUT>`

С помощью тега `<CFINPUT>` можно создавать такие управляющие элементы, как текстовые поля ввода, поля ввода пароля, флажки и переключатели (радиокнопки). За выбор типа управляющего элемента отвечает обязательный атрибут `TYPE` тега `<CFINPUT>`, который может принимать значения:

- ☐ `text` — текстовое поле ввода (по умолчанию);
- ☐ `password` — поле ввода пароля;
- ☐ `checkbox` — флажок;
- ☐ `radio` — переключатель.

Например:

```
<CFINPUT TYPE="text" NAME="FirstName" VALUE="Anna"
        SIZE="10" MAXLENGTH="11">
```

Таким образом, мы создаем текстовое поле с заранее определенным значением "Anna", указанием размера поля с помощью атрибута `SIZE` и установ-

кой максимального количества вводимых символов, используя атрибут MAXLENGTH.

Или, например:

```
<CFINPUT TYPE="password" NAME="Password"
        SIZE="10" MAXLENGTH="11">
```

С помощью приведенного примера создаем поле для ввода пароля. Следует отметить, что атрибуты SIZE и MAXLENGTH не распространяются на элементы с типами radio и checkbox.

Следующий пример:

```
<CFINPUT TYPE="checkbox" NAME="SubDis" VALUE="1" CHECKED="Yes">
```

Как видно по установленному атрибуту TYPE, это флажок, и определен он как выбранный по умолчанию, о чем говорит атрибут CHECKED. Причем передача указанного значения с помощью атрибута VALUE будет только в том случае, когда флажок отмечен как выбранный.

И еще один пример, демонстрирующий уже создание группы переключателей:

```
Children<CFINPUT TYPE="radio" NAME="AgeGroup" VALUE="1">
Adult<CFINPUT TYPE="radio" NAME="AgeGroup" VALUE="2" CHECKED>
Senior<CFINPUT TYPE="radio" NAME="AgeGroup" VALUE="3">
```

В этом примере отметим следующее, все три элемента имеют общее имя (атрибут NAME равен "AgeGroup"), тем самым мы указываем, что эти элементы относятся к одной группе, только один из них может быть выбран, и его значение будет передано. При использовании переключателей следует указывать значения элементов, используя атрибут VALUE. С помощью атрибута CHECKED мы указали выбранное по умолчанию значение равное 2. Заметим, что в ColdFusion для атрибутов с возможными значениями наподобие Yes/No, необязательно записывать Yes или No, можно просто указать атрибут, и такая конструкция будет означать Yes.

Полный код может выглядеть так:

```
<HTML>
<HEAD>
    <TITLE>CFINPUT</TITLE>
</HEAD>
<BODY>
    <h1>Log In</h1>
    <TABLE border='0' cellspacing='2' cellpadding='2'>
        <CFFORM NAME="Form1" ACTION="LogInGO.cfm">
            <TR><TD align="right">First Name:</TD><TD>
```



```

    <CFINPUT TYPE="text" NAME="FirstName" VALUE="Anna"
              SIZE="10" MAXLENGTH="11">
</TD></TR>
<TR><TD align="right">Password:</TD><TD>
    <CFINPUT TYPE="password" NAME="Password"
              SIZE="10" MAXLENGTH="11">
</TD></TR>
<TR><TD align="right">Age Group:</TD><TD>
    Children<CFINPUT TYPE="radio" NAME="AgeGroup" VALUE="1">
    Adult<CFINPUT TYPE="radio" NAME="AgeGroup" VALUE="2" CHECKED>
    Senior<CFINPUT TYPE="radio" NAME="AgeGroup" VALUE="3">
</TD></TR>
<TR><TD align="right">To subscribe for dispatch:</TD><TD>
    <CFINPUT TYPE="checkbox" NAME="SubDis" VALUE="1" CHECKED="Yes">
</TD></TR>
<TR><TD align="right"><input type='submit' value='Send'></TD>
    <TD><input type='reset' value='Reset'></TD></TR>
</CFFORM>
</TABLE>
</BODY>
</HTML>

```

На рис. 12.1 показан результат приведенного примера.

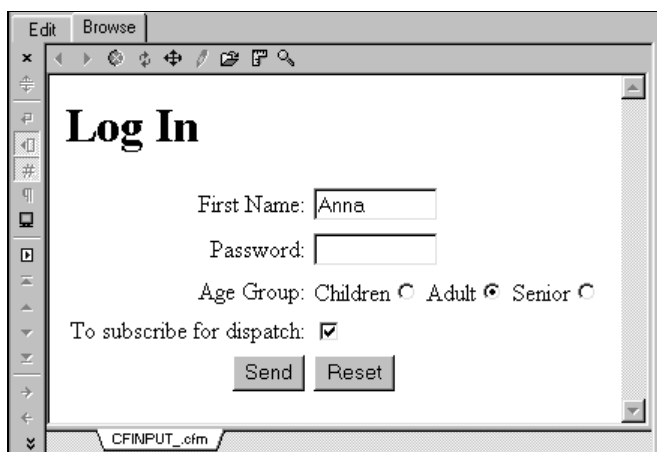


Рис. 12.1. Пример использования тега <CFINPUT>

Для удобства размещения элементов на странице мы использовали HTML-тег `<TABLE>`, а флажок `SubDis` поместили в конце для придания форме эстетического вида.

Контроль ввода данных

При создании форм достаточно часто приходится проверять правильность заполнения полей формы, как мы уже неоднократно демонстрировали, как с помощью программ на языке JavaScript, так и тегов ColdFusion. Однако при использовании тегов `<CFINPUT>` и `<CFTEXTINPUT>` можно проконтролировать введенные данные без написания какого-либо кода, указав атрибут `VALIDATE`, входящий в эти теги.

Атрибут `VALIDATE`, в свою очередь, может принимать следующие значения:

- ☐ `Date` — проверяет соответствие значения поля американскому формату даты (`mm/dd/yyyy`);
- ☐ `Eurodate` — проверяет соответствие значения поля европейскому формату даты (`dd/mm/yyyy`);
- ☐ `Time` — проверяет соответствие формату времени (`hh:mm:ss`);
- ☐ `Float` — проверяет соответствие типу данных `float`;
- ☐ `Integer` — проверяет соответствие типу данных `integer`;
- ☐ `Telephone` — проверяет соответствие телефонного номера формату `"###-###-####"`, где дефис можно опускать, а код города и телефонный номер должны начинаться с цифры;
- ☐ `Zipcode` — проверяет соответствие почтового адреса, который может состоять из 5 или 9 цифр, формату `"#####-####"`; только для США;
- ☐ `Creditcard` — проверяет соответствие номера кредитной карточки определенному формату;
- ☐ `Social_security_number` — проверяет соответствие номера социальной карточки формату `"###-##-####"`;
- ☐ `Regular_expression` — проверяет соответствие указанного выражения формату регулярного выражения.

При указании одного из вышеперечисленных значений атрибута `VALIDATE` также можно установить атрибут сообщения `MESSAGE`, которое будет выводиться на экран при заполнении данными недопустимого формата.

Контроль с помощью регулярных выражений

Помимо значений, контролирующих ввод данных с предопределенными правилами при использовании тегов `<CFINPUT>` и `<CFTEXTINPUT>`, можно применять *регулярные выражения* (regular expression) для проверки соответ-

вия текста, вводимого пользователем, собственному образцу, указанному с помощью атрибута `PATTERN`. Для определения образца того или иного поля ввода необходимо использовать обычные символы в сочетании со специальными символами.

Регулярные выражения позволяют проверять текстовые поля ввода для разнообразных условий. Можно создавать регулярное выражение, которое соответствует только числам в определенном диапазоне. Можно объединять простые регулярные выражения в сложные критерии поиска, чтобы соответствовать сложным образцам, например, одного из слов с различными окончаниями. Также в регулярных выражениях можно использовать переменные и функции ColdFusion, т. к. ColdFusion Server вычисляет переменные и функции прежде, чем определяется регулярное выражение.

В первую очередь следует отметить, что при формировании образца с помощью регулярного выражения для отображения специальных символов как обычных, следует использовать левую наклонную черту `\` в качестве предшествующего этому символу. Например, для отображения знака `+` следует писать `\+`.

Специальными символами следует считать:

`+ * ? . [ ^ $ ( ) { | \`

Приведем характерные правила для регулярных выражений.

- ☐ Точка (`.`) соответствует любому символу.
- ☐ Набор символов, указанных в квадратных скобках (`[]`), соответствует одному из символов в этих скобках. Например, `[abc]` предполагает `a`, `b` или `c`.
- ☐ Символ дефиса (`-`) позволяет указывать диапазон. Например, `[A-F]` предполагает любой символ от `A` до `F` в верхнем регистре.
- ☐ Если первым символом в скобках указан знак `^`, это соответствует любому кроме указанных символов. Например, `[^abc]` предполагает любой символ за исключением `a`, `b` или `c`.
- ☐ Для игнорирования чувствительности к регистру клавиатуры можно указывать необходимый символ двойным написанием как, например `[Aa]`.
- ☐ Группа символов, указанных в круглых скобках, представляет некую единицу, с которой можно производить какие-либо действия. Например, `(Ku)+` предполагает повторение `Ku`.
- ☐ Символы или сгруппированное выражение, сопровождаемое знаком звездочка (`*`), предусматривают наличие или отсутствие значения. Например, `[A-F]*` предполагает существование или отсутствие любого символа от `A` до `F` в верхнем регистре.
- ☐ Символы или сгруппированное выражение, сопровождаемые знаком вопроса (`?`), предполагают нулевое значение или регулярное выражение. Например `(good)?bye` предполагает `goodbye` или `bye`.

- ❑ Знак `^` в начале регулярного выражения соответствует началу поля.
- ❑ Знак `$` в конце регулярного выражения соответствует окончанию поля.
- ❑ Регулярное выражение может состоять из нескольких выражений. Например, `[A-F][a-f]*` предполагает любое слово, состоящие из букв от `a` до `f` независимо от регистра.
- ❑ Символ `|` позволяет осуществлять выбор между двумя выражениями. Например, `cold(er|est)` предполагает `colder` или `coldest`.
- ❑ Фигурные скобки `{ }` используются для указания диапазона повторения регулярного выражения в формате `{min,max}`, где `min` — положительное целое число, указывающее минимальное количество повторений, а `max` — число, равное или превосходящее `min`, указывает максимальное количество повторений. Например `(mu){2,4}` предполагает от 2 до 4 пар повторения `mu`. Синтаксис `{,max}` недопустим.

Также при формировании регулярных выражений можно использовать escape-последовательности. Например, такие как:

- ❑ `\d` — число от 0 до 9, соответствует `[0-9]`;
- ❑ `\D` — любой символ кроме чисел, соответствует `[^0-9]`;
- ❑ `\w` — алфавитно-числовой символ или подчеркивание, соответствует `"[A-Za-z0-9_]"`;
- ❑ `\W` — не алфавитно-числовой символ или подчеркивание, соответствует `"[^A-Za-z0-9_]"`

и т. д.

Пример использования атрибута **VALIDATE**

Приведенный ниже пример позволяет продемонстрировать проверку заполняемых полей текущей формы с помощью атрибута `VALIDATE`, имеющего как конкретные значения, так и значения на основе регулярного выражения, образец которого определен с помощью атрибута `PATTERN`:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>CFFORM Validation</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<h1>Sign Up</h1>
```

```
<TABLE border='0' cellpadding='2' cellspacing='0'>
```

```
<CFFORM NAME="Form1" ACTION="SignUpGO.cfm" >
```

```
<TR><TD align="right">E-Mail</TD><TR>
```

```
<CFINPUT TYPE="text" NAME="EMail"
```

```

        REQUIRED="Yes" VALIDATE="Regular_expression"
        PATTERN="^[A-Za-z0-9_.]{1}[A-Za-z0-9_.]+$"
        SIZE="15" MAXLENGTH="16">
</TD></TR>
<TR><TD align="right">First Name</TD><TD>
<CFINPUT TYPE="text" NAME="FirstName"
        REQUIRED="Yes" VALIDATE="Regular_expression"
        PATTERN="[A-Za-z]"
        MESSAGE="First Name, should not be empty and
                to include symbols 'A-Za-z'"
        SIZE="10" MAXLENGTH="11">
</TD></TR>
<TR><TD align="right">Birthday</TD><TD>
<CFINPUT TYPE="text" NAME="Birthday"
        REQUIRED="Yes" VALIDATE="Eurodate"
        SIZE="10" MAXLENGTH="11">
</TD></TR>
<TR><TD align="right">Phone Num.</TD><TD>
<CFINPUT TYPE="text" NAME="PhoneNum"
        REQUIRED="Yes" VALIDATE="Telephone"
        SIZE="12" MAXLENGTH="13">
</TD></TR>
<TR><TD align="right"><input type='submit' value=' Add '></TD>
        <TD><input type='reset' value='Clear'></TD></TR>
</CFFORM>
</table>
</BODY>
</HTML>

```

В приведенном примере мы создаем правила ввода для полей EMail и FirstName, используя регулярные выражения, и указываем контролирующие значения для полей Birthday и PhoneNum со встроенными правилами. При этом для поля FirstName создаем собственное сообщение об ошибке.

Контроль с помощью кода JavaScript

Как отмечалось, в предыдущих разделах мы уже применяли HTML-тег <FORM>, позволяющий создавать формы для обработки данных, где, впрочем, уже и использовали код JavaScript для контроля вводимых данных. При указании тега <CFFORM> также можно добавлять код JavaScript. Для этой цели

следует использовать атрибут `ONVALIDATE` таких элементов формы как `<CFINPUT>`, `<CFTEXTINPUT>`, `<CFSLIDER>`, `<CFTREE>`, `<CFGREED>`.

При этом функции, указанной атрибутом `ONVALIDATE`, передаются такие объекты, как форма (`form_obj`), текущий элемент ввода (`input_obj`) и значение текущего элемента ввода.

Можно также использовать атрибут `ONERROR` для указания функции JavaScript, которая выполняется, если при заполнении полей формы происходит ошибка, т. е. когда введенные значения не удовлетворяют установленным правилам, причем независимо от того, с помощью `VALIDATE` или `ONVALIDATE` регулируются эти правила.

Атрибут `ONERROR` возможно применять в таких элементах формы, как `<CFINPUT>`, `<CFTEXTINPUT>`, `<CFSELECT>`, `<CFSLIDER>`, `<CFTREE>`, `<CFGREED>`. При этом функции, указанной атрибутом `ONERROR`, передаются такие объекты, как форма (`form_obj`), текущий элемент ввода (`input_obj`), значение текущего элемента ввода и текст сообщения об ошибке.

Пример с использованием JavaScript

Приведенный далее пример продемонстрирует проверку заполняемых полей текущей формы, используя как значения атрибута `VALIDATE` с применением атрибута `ONERROR`, так и код JavaScript, указывая при этом названия функций с помощью атрибута `ONVALIDATE`:

```
<HTML>
<HEAD>
  <TITLE>CFFORM JavaScript Validation</TITLE>
</HEAD>
<BODY>
  <script>
    function Birthday_error(form, input, val, mess){
      input.value='dd/mm/yyyy';
      alert(mess);
    }
    function Pass_(form){
      if (form.Pass.value == "") { return (false); }
      else { return (true); }
    }
    function Conf_(form, input, val){
      if (form.Pass.value == val) { return (true); }
      else { return (false); }
    }
  </script>

```

```

</script>
<h1>Sign Up</h1>
<TABLE border='0' cellspacing='2' cellpadding='0'>
  <CFFORM NAME="Form1" ACTION="SignUpGO.cfm" >
    <TR><TD align="right">First Name</td><td>
      <CFINPUT TYPE="text" NAME="FirstName"
        REQUIRED="Yes" VALIDATE="Regular_expression"
        PATTERN="[A-Za-z]"
        SIZE="10" MAXLENGTH="11">
    </TD></TR>
    <TR><TD align="right">Birthday</TD><TD>
      <CFINPUT TYPE="text" NAME="Birthday"
        REQUIRED="Yes" VALIDATE="Eurodate"
        ONERROR="Birthday_error"
        SIZE="10" MAXLENGTH="11">
    </TD></TR>
    <TR><TD align="right">Password</TD><TD>
      <CFINPUT TYPE="password" NAME="Pass"
        REQUIRED="Yes" ONVALIDATE="Pass_"
        MESSAGE="Password, should not be empty"
        SIZE="8" MAXLENGTH="9">
    </TD></TR>
    <TR><TD align="right">Confirm</TD><TD>
      <CFINPUT TYPE="password" NAME="Conf"
        REQUIRED="Yes" ONVALIDATE="Conf_"
        MESSAGE="Fields: Password and Confirm,
          should be identical!"
        SIZE="8" MAXLENGTH="9">
    </TD></TR>
    <TR><TD align="right"><input type='submit' value=' Add '></TD>
      <TD><input type='reset' value='Clear'></TD></TR>
  </CFFORM>
</TABLE>
</BODY>
</HTML>

```

В приведенном примере мы создаем правила ввода для поля `FirstName`, используя регулярное выражение, указываем контролирующее значение для

поля Birthday со встроенным правилом с помощью атрибута ONERROR и останавливаем для полей Pass и Conf контроль над вводом данных с помощью JavaScript.

В то же время вы можете добавить HTML-тег <FORM> в комбинации с тегом <CFFORM>, используя атрибут PASSTHROUGH следующим образом:

```
<HTML>
<HEAD>
  <TITLE>CFFORM(PASSTHROUGH) JavaScript Validation</TITLE>
</HEAD>
<BODY>
  <script>
    function FName(val){
      if(val=="") return false;
      else return true;
    }
  </script>
  <h1>Sign Up</h1>
  <TABLE border='0' cellspacing='2' cellpadding='0'>
    <CFFORM NAME="Form1" ACTION="SignUpGO.cfm">
      <TR><TD align="right">First Name</TD><TD>
        <CFINPUT TYPE="text" NAME="FirstName"
          REQUIRED="Yes"
          SIZE="10" MAXLENGTH="11"
          MESSAGE="First Name, should not be empty"
          PASSTHROUGH="Yes" onClick = "FName(this.value)">
      </TD></TR>
      <TR><TD align="right"><input type='submit' value=' Add '></TD>
      <TD><input type='reset' value='Clear'></TD></TR>
    </CFFORM>
  </TABLE>
</BODY>
</HTML>
```

В приведенном примере мы указываем атрибут PASSTHROUGH для поля FirstName и определяем при возникновении события onClick выполнение функции FName(this.value), что позволяет контролировать ввод данных для этого поля с помощью JavaScript.

Тег <CFTEXTINPUT>

Тег <CFTEXTINPUT> позволяет создавать текстовые поля ввода с возможностью форматирования текста. Этот управляющий элемент реализован на основании Java-апплета.

Например:

```
<CFTEXTINPUT NAME="textblock1"
    HEIGHT="80" WIDTH="400" ALIGN="Top"
    BGCOLOR="##8080C0" TEXTCOLOR="White"
    BOLD="Yes" ITALIC="Yes" FONTSIZE="72"
    VALUE="TextBlock1">
```

Здесь с помощью обязательного атрибута NAME задаем имя элемента <CFTEXTINPUT>, указываем высоту и ширину текстового поля в пикселах, используя атрибуты HEIGHT и WIDTH, задаем выравнивание поля по верхнему краю с помощью атрибута ALIGN, устанавливаем цвет заливки фона и цвет текста с помощью атрибутов BGCOLOR и TEXTCOLOR, придаем тексту жирный и наклонный характер, используя атрибуты BOLD и ITALIC, а также определяем размер шрифта атрибутом FONTSIZE и устанавливаем значение текстового поля атрибутом VALUE.

Заметим, что для назначения цвета необходимо использовать двойной символ # перед шестнадцатеричным числовым кодом или задавать цвет по названию, используя перечисленные ниже значения:

- ☐ Black — черный;
- ☐ Magenta — пурпурный;
- ☐ Cyan — голубой;
- ☐ Orange — оранжевый;
- ☐ Darkgray — темно-серый;
- ☐ Pink — розовый;
- ☐ Gray — серый;
- ☐ White — белый;
- ☐ Lightgray — светло-серый;
- ☐ Yellow — желтый.

И еще один пример из разряда применения тега <CFTEXTINPUT>:

```
<CFTEXTINPUT NAME="textblock2"
    HEIGHT="40" WIDTH="400" ALIGN="Bottom"
    BGCOLOR="##FFFFD5" TEXTCOLOR="Gray"
    FONTSIZE="36" VSPACE="5" FONT="Zurich Ex BT"
    VALUE="TextBlock2">
```

Здесь в отличие от предыдущего примера добавлены еще два атрибута: `VSPACE`, позволяющий устанавливать величину незаполненного пространства в пикселах сверху и снизу от содержащего данный атрибут элемента, и `FONT`, предоставляющий возможность указывать шрифт текста.

Включим описанные примеры в одну страницу следующим образом:

```
<HTML>
<HEAD>
  <TITLE>CFTEXTINPUT</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFFORM NAME="Form1" ACTION="CFTEXTINPUT_.cfm">
    <CFTEXTINPUT NAME="textblock1"
      HEIGHT="80" WIDTH="400" ALIGN="Top"
      BGCOLOR="##8080C0" TEXTCOLOR="White"
      BOLD="Yes" ITALIC="Yes" FONTSIZE="72"
      VALUE="TextBlock1">
    <CFTEXTINPUT NAME="textblock2"
      HEIGHT="40" WIDTH="400" ALIGN="Bottom"
      BGCOLOR="##FFFFFFD5" TEXTCOLOR="Gray"
      FONTSIZE="36" VSPACE="5"
      FONT="Zurich Ex BT"
      VALUE="TextBlock2">
  </CFFORM>
</BODY>
</HTML>
```

На рис. 12.2 показан результат приведенного примера.

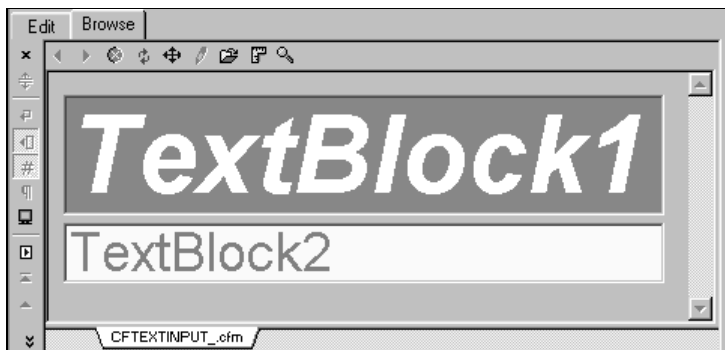


Рис. 12.2. Пример использования тега `<CFTEXTINPUT>`

Мы не обсуждаем все существующие атрибуты тега `<CFTEXTINPUT>` и их возможные значения детально, но уверяем, их осталось не так много. Тег `<CFTEXTINPUT>`, так же, как и тег `<CFINPUT>`, поддерживает атрибуты `MESSAGE`, `ONERROR`, `REQUIRED`, `VALIDATE` и `ONVALIDATE`.

Тег `<CFSELECT>`

Тег `<CFSELECT>` задает список, позволяющий выбирать один или несколько вариантов из списка. Если атрибут `MULTIPLE` не указан, то создается простой раскрывающийся список, в котором можно выбирать только один вариант. Его значение всегда передается, т. к. всегда хотя бы один пункт выбран. При указании атрибута `MULTIPLE` можно выбирать несколько значений. Синтаксис данного тега аналогичен синтаксису HTML-тега `<select>`.

Например:

```
<CFSELECT NAME="Destination1">
  <OPTION VALUE="1">World Trade Center
  <OPTION VALUE="2">World Financial Center
  <OPTION VALUE="3">Museum of Jewish Heritage
  <OPTION VALUE="4">Battery Park
</CFSELECT>
```

Здесь с помощью обязательного атрибута `NAME` задаем наименование элемента `<CFSELECT>` и указываем опции списка, используя знакомую конструкцию. Во всяком случае, эта конструкция хорошо знакома тем, кому доводилось применять HTML-тег `<select>`.

Так же, как и при использовании HTML-тега `<select>`, можно устанавливать значение по умолчанию, добавив при этом атрибут `SELECTED` для конкретной опции. Например:

```
<CFSELECT NAME="Destination2" SIZE="3">
  <OPTION VALUE="1">Ellis Island
  <OPTION VALUE="2">Wall Street
  <OPTION VALUE="3" SELECTED>Statue of Liberty
</CFSELECT>
```

В приведенном примере мы указали размер видимой части списка с помощью атрибута `SIZE`, однако это еще не повод, чтобы выбирать сразу несколько опций, удерживая клавиши `<Ctrl>` и `<Shift>`. Для подобной возможности, как было сказано, необходимо добавить атрибут `MULTIPLE`. Например:

```
<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
  SELECT Key_tou, Name_tou
```

```

FROM Tour
</CFQUERY>
<CFSELECT NAME="Tour1" MULTIPLE="Yes" SIZE="3">
  <CFOUTPUT QUERY="GetTour">
    <OPTION VALUE="#key_tou#">#name_tou#
  </CFOUTPUT>
</CFSELECT>

```

В этом фрагменте кода в первую очередь следует обратить внимание на предварительно построенный запрос с помощью тега `<CFQUERY>`, который обсуждался в предыдущей главе. Это позволяет нам определять значения опций из базы данных. Однако для использования данных запроса в теге `<CFSELECT>` присутствуют следующие атрибуты: `QUERY` — имя запроса, `VALUE` — колонка запроса, выступающая в качестве значений списка, и `DISPLAY` — колонка запроса, отображаемая в списке. Например:

```

<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
  SELECT Key_tou, Name_tou
  FROM Tour
</CFQUERY>
<CFSELECT NAME="Tour2" MULTIPLE="Yes" SELECTED="3"
  QUERY="GetTour" VALUE="key_tou" DISPLAY="name_tou">
</CFSELECT>

```

Атрибут `SELECTED` мы добавили исключительно для демонстрации указания значения по умолчанию.

Включим описанные примеры в одну страницу следующим образом:

```

<HTML>
  <HEAD>
    <TITLE>CFSELECT</TITLE>
  </HEAD>
  <BODY bgcolor="Silver">
    <CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
      SELECT Key_tou, Name_tou
      FROM Tour
    </CFQUERY>
    <CFFORM NAME="Form1" ACTION="CFSELECT_.cfm">
      <CFSELECT NAME="Destination1">
        <OPTION VALUE="1">World Trade Center
        <OPTION VALUE="2">World Financial Center
        <OPTION VALUE="3">Museum of Jewish Heritage
        <OPTION VALUE="4">Battery Park
      </CFSELECT>
    </CFFORM>
  </BODY>
</HTML>

```

```
</CFSELECT>
<br>
<CFSELECT NAME="Destination2" SIZE="3">
  <OPTION VALUE="1">Ellis Island
  <OPTION VALUE="2">Wall Street
  <OPTION VALUE="3" SELECTED>Statue of Liberty
</CFSELECT>
<br>
<CFSELECT NAME="Tour1" MULTIPLE="Yes" SIZE="3">
  <CFOUTPUT QUERY="GetTour">
    <OPTION VALUE="#key_tou#">#name_tou#
  </CFOUTPUT>
</CFSELECT>
<br>
<CFSELECT NAME="Tour2" MULTIPLE="Yes" SELECTED="3"
  QUERY="GetTour" VALUE="key_tou" DISPLAY="name_tou">
</CFSELECT>
</CFFORM>
</BODY>
</HTML>
```

На рис. 12.3 показан результат приведенного примера.

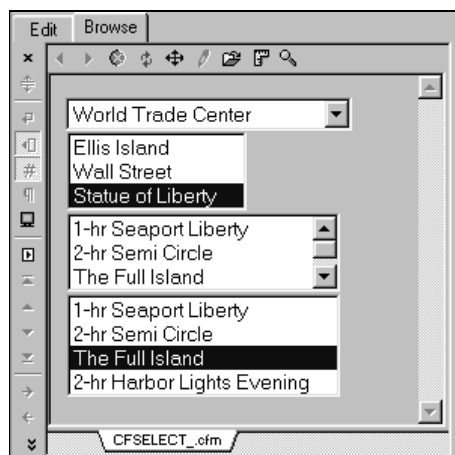


Рис. 12.3. Пример использования тега <CFSELECT>

Тег <CFSELECT>, так же, как теги <CFINPUT> и <CFTEXTINPUT>, поддерживает атрибуты MESSAGE, ONERROR и REQUIRED, и, кроме того, атрибут PASSTHROUGH.

Тег <CFSLIDER>

Тег <CFSLIDER> позволяет разместить в форме движок со шкалой или без нее, перемещаемый по вертикали или горизонтали и предоставляющий возможность визуально управлять числовыми величинами в указанном диапазоне. Этот управляющий элемент реализован на основании Java-апплета.

Например:

```
<CFSLIDER NAME="my1" RANGE="0,100" SCALE="1" VALUE="33" HEIGHT="25">
```

Здесь с помощью обязательного атрибута NAME задаем имя элемента <CFSLIDER>, определяем диапазон значений движка атрибутом RANGE, указываем масштаб шкалы движка с помощью атрибута SCALE, а также устанавливаем значение по умолчанию, используя атрибут VALUE, и указываем ширину горизонтального движка в пикселах атрибутом HEIGHT.

По умолчанию значение диапазона, устанавливаемого с помощью атрибута RANGE, равно от 0 до 100, а значение шкалы движка SCALE равно 1. То есть приведенный выше пример можно видоизменить, и результат при этом будет тождественен:

```
<CFSLIDER NAME="my1" VALUE="33" HEIGHT="25">
```

При размещении движка можно добавить подпись (label) с отображением текущего значения данного управляющего элемента, как показано в следующем примере:

```
<CFSLIDER NAME="my2" VALUE="50"  
    VSPACE="5" BGCOLOR="Gray" TEXTCOLOR="###FFFFD5"  
    LABEL="Value Slider: " WIDTH="200" HEIGHT="45">
```

Здесь помимо атрибута LABEL для отображения подписи задаются уже известные атрибуты, используемые в описанных выше управляющих элементах.

С помощью атрибута TICKMARKLABELS можно включить отображение подписи шкалы движка с шагом, соответствующим масштабу шкалы, указанному атрибутом SCALE, как это показано в следующем примере:

```
<CFSLIDER NAME="my3" RANGE="0,50" SCALE="10" VALUE="10"  
    LABEL="Value Slider: " WIDTH="200" HEIGHT="60"  
    TICKMARKLABELS="Yes">
```

Также можно включать отображение самой шкалы, в виде линейки с засечками, добавив при этом атрибуты TICKMARKMAJOR или TICKMARKMINOR. И если в первом случае будут отображены только основные засечки в соответствии с установленным масштабом, то при использовании TICKMARKMINOR также будут показаны промежуточные засечки меньшей величины.

Например:

```
<CFSLIDER NAME="my4" RANGE="50,100" SCALE="10" VALUE="90"  
    HSPACE="5" WIDTH="200" HEIGHT="60"  
    TICKMARKMINOR="True" TICKMARKLABELS="Yes">
```

Здесь `HSPACE` позволяет устанавливать величину незаполненного пространства в пикселах слева и справа от содержащего данный атрибут элемента.

Если вас не устраивает, что в качестве подписи шкалы отображаются числовые величины в указанном диапазоне, то, используя атрибут `TICKMARKLABELS`, можно сформировать собственный список значений, который станет отображаться в виде подписей шкалы. Причем, если диапазон меньше указанного количества элементов списка, лишние элементы будут игнорироваться, если же, наоборот, диапазон будет превосходить количество элементов списка, то последний элемент повторится. Приведем пример со списком времен года:

```
<CFSLIDER NAME="my5" RANGE="1,4" VALUE="1"  
    VSPACE="5" LABEL="Seasons: " WIDTH="300" HEIGHT="70"  
    TICKMARKMINOR="True"  
    TICKMARKLABELS="Winter, Spring, Summer, Autumn"  
    FONT="Impact" BOLD="Yes" FONTSIZE="14">
```

В этом фрагменте добавлены атрибуты `FONT`, `BOLD` и `FONTSIZE`, позволяющие форматировать подпись, указанную атрибутом `LABEL`.

Все предыдущие примеры демонстрировали движки в горизонтальном положении. Если же есть необходимость изобразить движок вертикально, то достаточно включить атрибут `VERTICAL`, как показано в следующем примере:

```
<CFSLIDER NAME="my6" RANGE="1,5" VALUE="1"  
    WIDTH="120" HEIGHT="120" VERTICAL="True"  
    TICKMARKMINOR="True" LABEL=" Language: "  
    TICKMARKLABELS="Russian, English, German, Spanish, French"  
    TICKMARKIMAGES="rus.gif, eng.gif, ger.gif, spa.gif, fre.gif">
```

Здесь присутствует еще один интересный атрибут, `TICKMARKIMAGES`, предполагающий перечисление через запятую графических файлов, содержимое которых в свою очередь будет выведено между шкалой и подписью шкалы. При этом, так же, как и с атрибутом `TICKMARKLABELS`, если диапазон меньше указанного количества графических файлов, лишние рисунки будут игнорироваться, если же, наоборот, диапазон будет превосходить количество графических файлов, то последний рисунок, указанный с помощью атрибута `TICKMARKIMAGES`, повторится *n* раз.

Сведем все воедино:

```

<HTML>

<HEAD>

    <TITLE>CFSLIDER</TITLE>

</HEAD>

<BODY bgcolor="Silver">

    <CFFORM NAME="Form1" ACTION="CFSLIDER_.cfm">

        <CFSLIDER NAME="my1" RANGE="0,100" SCALE="1" VALUE="33"
            HEIGHT="25">

        <br>

        <CFSLIDER NAME="my2" VALUE="50"
            VSPACE="5" BGCOLOR="Gray" TEXTCOLOR="##FFFFD5"
            LABEL="Value Slider: " WIDTH="200" HEIGHT="45">

        <br>

        <CFSLIDER NAME="my3" RANGE="0,50" SCALE="10" VALUE="10"
            LABEL="Value Slider: " WIDTH="200" HEIGHT="60"
            TICKMARKLABELS="Yes">

        <CFSLIDER NAME="my4" RANGE="50,100" SCALE="10" VALUE="90"
            HSPACE="5" WIDTH="200" HEIGHT="60"
            TICKMARKMAJOR="True" TICKMARKMINOR="True"
            TICKMARKLABELS="Yes">

        <br>

        <CFSLIDER NAME="my5" RANGE="1,4" VALUE="1"
            VSPACE="5" LABEL="Seasons: " WIDTH="300" HEIGHT="70"
            TICKMARKMINOR="True"
            TICKMARKLABELS="Winter, Spring, Summer, Autumn"
            FONT="Impact" BOLD="Yes" FONTSIZE="14">

        <br>

        <CFSLIDER NAME="my6" RANGE="1,5" VALUE="1"
            WIDTH="120" HEIGHT="120" VERTICAL="True"
            TICKMARKMINOR="True" LABEL=" Language: "
            TICKMARKLABELS="Russian, English, German, Spanish, French"
            TICKMARKIMAGES="rus.gif, eng.gif, ger.gif, spa.gif, fre.gif">

    </CFFORM>

</BODY>

</HTML>

```


На рис. 12.4 показан результат приведенного примера.

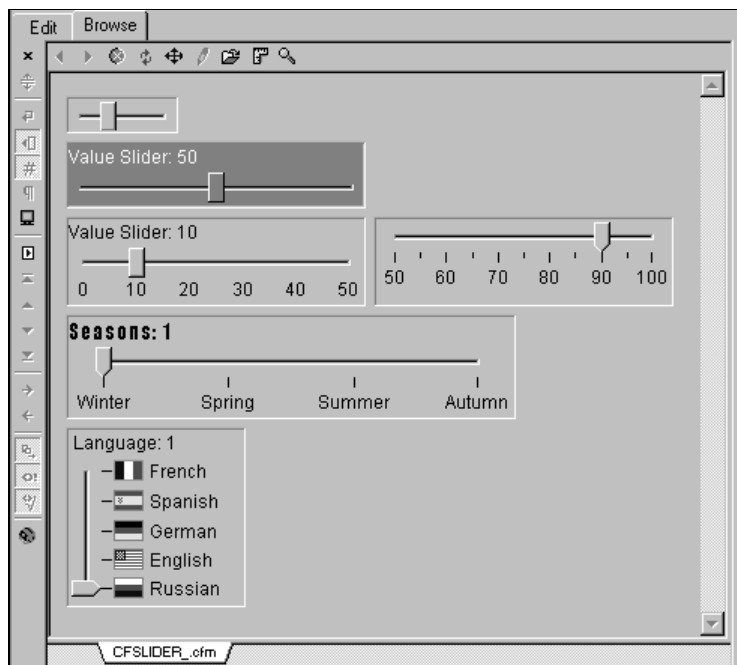


Рис. 12.4. Пример использования тега `<CFSLIDER>`

Стоит отметить еще один интересный атрибут управляющего элемента `<CFSLIDER>`. Это `LOOKANDFEEL`, который применяется для задания стиля элементов и может принимать следующие значения:

- ☐ `Motif` — стиль `Motif`;
- ☐ `Windows` — стиль (по умолчанию), используемый `Windows`;
- ☐ `Metal` — стиль, используемый `Java Swing`.

Например, изменим уже представленные примеры, добавив стиль `Motif`, как это показано для одного из первых примеров:

```
<CFSLIDER NAME="my1" RANGE="0,100" SCALE="1" VALUE="33"
    HEIGHT="25" LOOKANDFEEL="Motif">
```

На рис. 12.5 приведен результат.

Тег `<CFSLIDER>`, так же, как теги `<CFINPUT>` и `<CFTEXTINPUT>`, поддерживает атрибуты `MESSAGE`, `ONERROR` и `ONVALIDATE`.

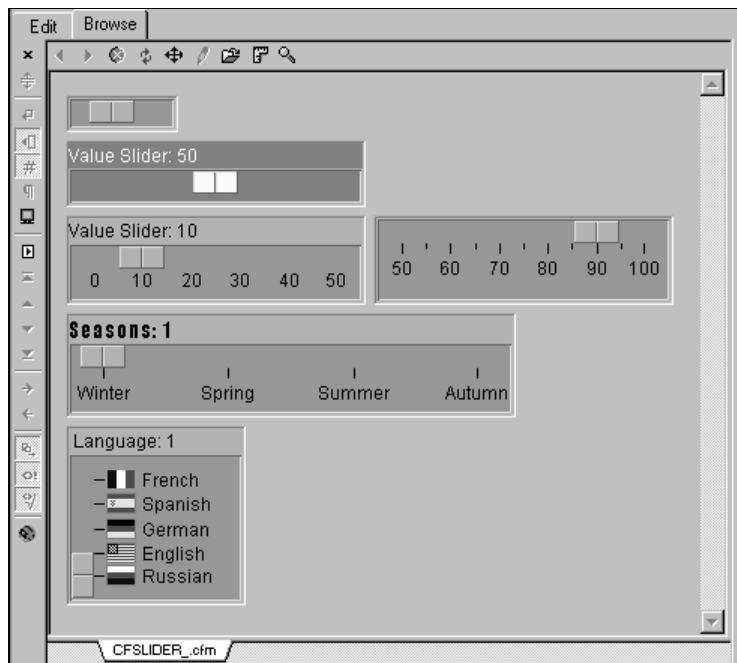


Рис. 12.5. Пример использования тега `<CFSLIDER>` в стиле Motif

Тег `<CFTREE>`

Тег `<CFTREE>` позволяет разместить в форме дерево объектов и представляет собой элемент управления для создания иерархического списка, в котором каждый пункт может иметь подчиненный пункт или набор пунктов, выводимых с отступом. Пользователь может сворачивать список, делая невидимым подчиненные пункты, или, наоборот, раскрывать иерархию управляющего элемента. Пункт дерева может содержать текст и изображение. Этот управляющий элемент реализован на основе Java-апплета.

Для реализации пунктов необходимо внутри тега `<CFTREE>` использовать тег `<CFTREEITEM>`. Например, представим код страницы без использования запроса:

```
<HTML>
<HEAD>
  <TITLE>CFTREE</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <h1>Structuring Tegn</h1>
```

```

<CFFORM NAME="Form1" ACTION="CFTREE_GO.cfm">
  <CFTREE NAME="StructuringTegs" HEIGHT="270" WIDTH="230"
    ALIGN="TOP" COMPLETEPATH="Yes">
    <CFTREEITEM VALUE="1.CFFORM">
    <CFTREEITEM VALUE="2.FORM">
    <CFTREEITEM VALUE="1.1.CFINPUT"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="1.2.CFTEXTINPUT"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="1.3.CFSELECT"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="1.4.CFSLIDER"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="1.5.CFTREE"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="1.6.CFGRID"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="1.7.CFAPPLET"
      PARENT="1.CFFORM" IMG="Element">
    <CFTREEITEM VALUE="2.1.INPUT"
      PARENT="2.FORM" IMG="Element">
    <CFTREEITEM VALUE="2.2.TEXTAREA"
      PARENT="2.FORM" IMG="Element">
    <CFTREEITEM VALUE="2.3.SELECT"
      PARENT="2.FORM" IMG="Element">
    <CFTREEITEM VALUE="1.5.1.CFTREEITEM"
      PARENT="1.5.CFTREE" IMG="Document">
    <CFTREEITEM VALUE="1.6.1.CFGRIDCOLUMN"
      PARENT="1.6.CFGRID" IMG="Document">
    <CFTREEITEM VALUE="1.6.2.CFGRIDROW"
      PARENT="1.6.CFGRID" IMG="Document">
  </CFTREE>
  <input type='submit' value='Send'>
</CFFORM>
</BODY>
</HTML>

```

Здесь обязательный атрибут `VALUE` тега `<CFTREEITEM>` задает значение пункта, а атрибут `PARENT` указывает на принадлежность к пункту более старшего

уровня иерархии или, как говорят, позволяет указать родителя. Атрибут `IMG` предоставляет возможность выбрать изображение для данного пункта, причем варианты могут быть следующие:

- ☐ `Cd` — компакт-диск;
- ☐ `Computer` — компьютер;
- ☐ `Document` — документ;
- ☐ `Element` — элемент;
- ☐ `Folder` — папка;
- ☐ `Floppy` — накопитель для гибких дисков;
- ☐ `Fixed` — подсоединенное устройство;
- ☐ `Remote` — сетевое устройство.

Впрочем, если ни один из существующих вариантов вас не устраивает, для атрибута `IMG` можно указывать путь к графическому файлу, который будет играть роль изображения для текущего пункта, как это будет показано ниже.

На рис. 12.6 представлен результат приведенного примера.

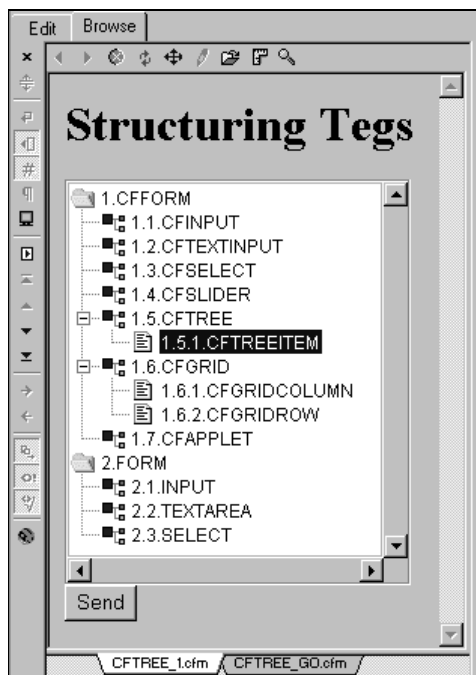


Рис. 12.6. Пример использования тега `<CFTREE>`

Управляющий элемент `<CFTREE>` позволяет выбирать один из пунктов иерархического списка, и при обработке формы передаются две переменные:

- ❑ `Form.treename.node` — возвращает значение выбранного пункта, узла;
- ❑ `Form.treename.path` — возвращает значение полного пути в иерархическом списке для выбранного пункта, в форме `[root]\node1\node2\node_n\value`.

Чтобы возвращать часть корня пути (`root`), необходимо установить атрибут `COMPLETEPATH` тега `<CFTREE>`, как показано в предыдущем примере. Причем, если пользователь выбирает пункт `"1.5.1.CFTREEITEM"`, в этом случае будут переданы переменные:

```
Form.StructuringTegs.node = 1.5.1.CFTREEITEM
```

```
Form.StructuringTegs.path = 1.CFFORM\1.5.CFTREE\1.5.1.CFTREEITEM
```

По умолчанию в качестве разделителя применяется левая наклонная черта. Однако, используя атрибут `DELIMITER` тега `<CFTREE>`, этот разделительный символ всегда можно изменить.

Следующим примером мы предвараем создание иерархического списка с использованием запросов и применением гипертекстовых ссылок:

```
<HTML>
<HEAD>
  <TITLE>CFTREE</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
    SELECT Key_tra, 'mailto:' + EMail_tra AS EMail,
           FirstName_tra + ' ' + LastName_tra AS TName
    FROM Traveler
    ORDER BY FirstName_tra, LastName_tra
  </CFQUERY>
  <h1>Travelers Tree</h1>
  <CFFORM NAME="Form1" ACTION="CFTREE_2.cfm">
    <CFTREE NAME="Tour" HEIGHT="290" WIDTH="190"
           HSCROLL="No" FONTSIZE="13" FONT="Trebuchet MS">
      <CFTREEITEM VALUE="Key_tra" DISPLAY="TName"
                  QUERY="GetTraveler" QUERYASROOT="Travelers"
                  IMG="People.gif,BulletCircle.gif" HREF="",EMail">
    </CFTREE>
  </CFFORM>
</BODY>
</HTML>
```

Здесь предварительно создается запрос с необходимыми данными, после чего указывается обязательный атрибут `VALUE` и имя запроса с помощью атрибута `QUERY` тега `<CFTREEITEM>`. В качестве атрибута `VALUE` задается одно или несколько полей запроса, перечисленных через запятую. Причем каждое последующее указанное поле будет представляться в качестве вложенного пункта в предыдущее поле. Если необходимо в списке отобразить одни поля базы данных, а в качестве передаваемых значений использовать другие, то для этой цели следует добавить атрибут `DISPLAY`.

Атрибут `QUERYASROOT` указывает на использование имени запроса как верхнего уровня иерархического списка и может принимать следующие значения:

- ☐ Yes — да;
- ☐ No — нет, при этом элементы запроса будут являться верхним уровнем иерархического списка;
- ☐ собственный текст наименования корня, как показано в примере.

На рис. 12.7 приведен результат выполнения этого примера.

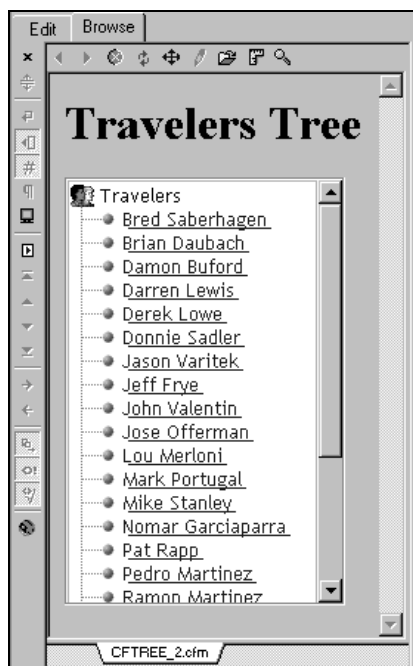


Рис. 12.7. Пример использования тега `<CFTREE>` с применением гипертекстовых ссылок

Еще одной из интересных деталей приведенного примера является использование атрибута `HREF`, позволяющего представлять надписи пунктов списка

в виде гипертекстовых ссылок. Как можно было заметить, в нашем примере мы извлекаем из базы данных почтовые адреса путешественников и добавляем в начало каждого адреса строку "mailto:", что даст возможность пользователю отправлять письма по адресу того или иного путешественника. Заметим, что в нашем примере атрибут `href` указывает на наличие гипертекстовых ссылок для второго уровня иерархического списка, о чем свидетельствует наличие запятой. Также можно напрямую указывать ссылку на вызываемый ресурс, например:

```
href = "http://www.youserver.com"
```

При этом, когда пользователь выбирает один из пунктов иерархического списка, чтобы активизировать данный ресурс, то в конец URL добавляется строка, позволяющая передавать значение выбранного пункта в качестве переменной по следующему правилу:

```
http://www.youserver.com?CFTREEITEMKEY=selected_value
```

Для отключения добавления переменной в конец URL следует при использовании тега `<CFTREE>` установить атрибут `appendkey="no"`. Например, несколько видоизменим приведенный выше пример:

```
<CFFORM NAME="Form1" ACTION="CFTREE_2.cfm">
  <CFTREE NAME="Tour" HEIGHT="290" WIDTH="190"
    HSCROLL="no" FONTSIZE="13" FONT="Trebuchet MS" APPENDKEY="no">
    <CFTREEITEM VALUE="Key_tra" DISPLAY="TName"
      QUERY="GetTraveler" QUERYASROOT="Travelers"
      IMG="People.gif,BulletCircle.gif"
      href="http://localhost,EMail" TARGET="_blank">
    </CFTREE>
  </CFFORM>
```

В данном фрагменте был включен еще один атрибут, `target`, позволяющий указывать правило открытия целевой страницы. В нашем случае значение `_blank` открывает по ссылке страницу в новом окне обозревателя. Так же, используя этот атрибут, можно задавать имя фрейма. Заметим, что мы указываем правило открытия страницы только для первого уровня иерархического списка. В этом же примере мы устанавливаем следующие атрибуты: `hscroll`, который отвечает за отображение горизонтальной полосы прокрутки, `fontsize` и `font`, которые отвечают за форматирование текста, и уже упомянутый атрибут `img`, предоставляющий возможность указать изображение для каждого уровня иерархического списка.

Представим еще один пример создания иерархического списка с использованием запроса:

```
<html>
<head>
```

```

<TITLE>CFTREE</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFQUERY NAME="GetRoute" DATASOURCE="#application.dsn#">
    SELECT Key_rou, Name_tou, Name_des
    FROM Tour, Destination, Route
    WHERE Route.key_tou=Tour.key_tou
    AND Route.key_des=Destination.key_des
  </CFQUERY>
  <h1>Routes Tree</h1>
  <CFFORM NAME="Form1" ACTION="CFTREE_3.cfm">
    <CFTREE NAME="Route" HEIGHT="250" WIDTH="260" HSCROLL="No">
      <CFTREEITEM VALUE="Name_tou, Name_des" QUERY="GetRoute"
        EXPAND="No" IMG="bulletL.gif,bullet.gif"
        IMGOPEN="bulletL2.gif">
    </CFTREE>
  </CFFORM>
</BODY>
</HTML>

```

Здесь с помощью атрибута `VALUE` указаны два поля, данные первого из которых будут представлены как основной уровень, а данные второго будут включены в качестве подчиненного уровня иерархического списка. В этом примере мы использовали атрибут `IMGOPEN`, позволяющий назначить изображение для пункта, находящегося в раскрытом состоянии, причем данный атрибут может принимать значения, аналогичные атрибуту `IMG`. Атрибут `EXPAND`, также примененный в последнем примере, позволяет определить состояние пунктов: в раскрытом виде с отображаемыми подпунктами — значение `Yes` (по умолчанию), в скрытом с невидимыми подпунктами — значение `No`.

На рис. 12.8 показан результат приведенного примера.

Отметим, что так же, как была скрыта горизонтальная полоса прокрутки управляющего элемента `<CFTREE>` с использованием атрибута `HSCROLL`, можно скрыть вертикальную полосу прокрутки, указав атрибут `VSCROLL`. Управляющий элемент `<CFTREE>`, так же, как и `<CFSLIDER>`, поддерживает использование атрибута `LOOKANDFEEL`, который позволяет задавать один из представленных стилей, `Motif`, `Windows` (по умолчанию) или `Metal`.

Тег `<CFTREE>` подобно тегам `<CFINPUT>`, `<CFTEXTINPUT>` и `<CFSLIDER>` поддерживает атрибуты `MESSAGE`, `ONERROR` и `ONVALIDATE`.

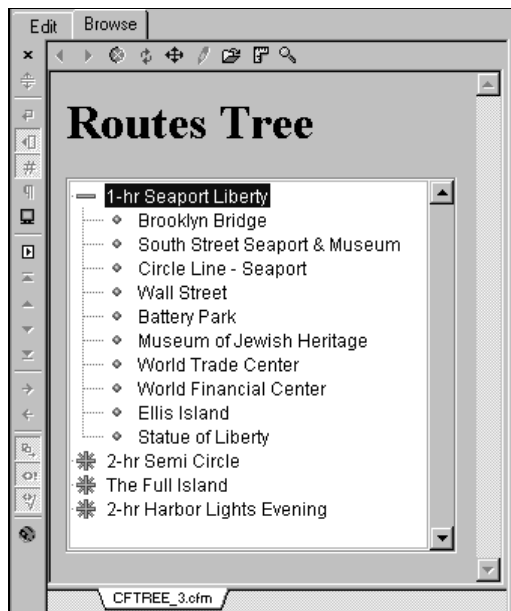


Рис. 12.8. Пример использования тега `<CFTREE>` с применением запроса

Тег `<CFGRID>`

Тег `<CFGRID>` отображает сетку для данных, позволяющую организовать работу с данными в виде таблицы со строками и колонками. Этот управляющий элемент реализован на основе Java-апплета.

Управляющий элемент `<CFGRID>` позволяет:

- ☐ размещать в таблице графические изображения;
- ☐ подключать данные, используя запросы к базам данных;
- ☐ сортировать включенные данные;
- ☐ модифицировать, добавлять и удалять данные;
- ☐ визуально изменять ширину и порядок отображения колонок;
- ☐ видоизменять фон и шрифт таблицы;
- ☐ управлять отображением заголовка, разделительных линий, колонки с порядковыми номерами;
- ☐ представлять надписи командных кнопок в виде изображений;
- ☐ выводить данные различными цветами в зависимости от их значения.

Для управления колонками и строками в <CFGRID> необходимо использовать теги <CFGRIDCOLUMN> и <CFGRIDROW>. Например, создадим сетку с четырьмя колонками и двумя строками:

```
<HTML>
<HEAD>
  <TITLE>CFGRID</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <h1>Grid</h1>
  <CFFORM NAME="Form1" ACTION="CFGRID_1.cfm">
    <CFGRID NAME="myGrid"
      WIDTH="240" HEIGHT="90" SELECTMODE="Single">
      <CFGRIDCOLUMN NAME="One">
      <CFGRIDCOLUMN NAME="Two">
      <CFGRIDCOLUMN NAME="Three">
      <CFGRIDCOLUMN NAME="Pic" TYPE="Image">
      <CFGRIDROW DATA="1,2,3,bulletL.gif">
      <CFGRIDROW DATA="4,5,6,bulletL2.gif">
    </CFGRID>
  </CFFORM>
</BODY>
</HTML>
```

Здесь четвертая колонка представляет графические изображения, т. к. с помощью атрибута TYPE тега <CFGRIDCOLUMN> указан соответствующий тип. Типы данных могут быть следующими:

- ☐ Image — изображение;
- ☐ Numeric — числовые данные;
- ☐ Boolean — при использовании данного типа ячейки отображаются в виде флажка (checkbox), предназначенного для данных с логическим значением;
- ☐ String_noCase — колонка с данными без учета чувствительности к регистру при использовании сортировки данных.

В свою очередь, сами изображения указаны как ссылка на графические файлы. Впрочем, здесь можно назначать уже известные изображения такие как Cd (компакт-диск), Computer (компьютер), Document (документ), Element (элемент), Folder (папка), Floppy (накопитель для гибких дисков), Fixed (подсоединенное устройство), Remote (сетевое устройство).

На рис. 12.9 показан результат приведенного примера.

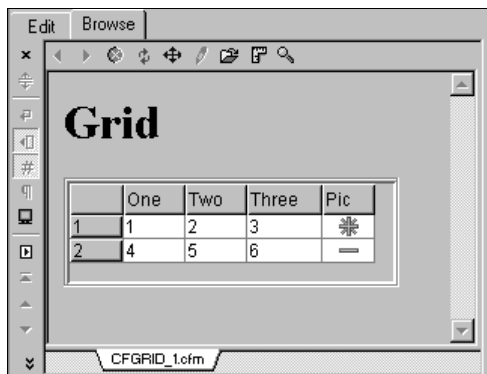


Рис. 12.9. Пример использования тега `<CFGRID>` с графическими изображениями

Немаловажным является атрибут `SELECTMODE` тега `<CFGRID>`, предназначенный для выбора способа отражения и поведения данных, который может принимать следующие значения:

- ☐ `Browse` — только для просмотра, выделение ячеек отключено (по умолчанию);
- ☐ `Single` — при перемещении по управляющему элементу выделяется текущая ячейка;
- ☐ `Row` — при перемещении выделяется вся текущая строка;
- ☐ `Column` — при перемещении выделяется вся текущая колонка;
- ☐ `Edit` — позволяет редактировать данные.

Приведенный ниже пример интересен тем, что данные элемента `<CFGRID>` определяются запросом к базе данных при использовании атрибута `QUERY`:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>CFGRID</TITLE>
```

```
</HEAD>
```

```
<BODY bgcolor="Silver">
```

```
<CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
```

```
SELECT *
```

```
FROM Traveler
```

```
ORDER BY FirstName_tra, LastName_tra
```

```
</CFQUERY>
```

```
<h1>Travelers Grid</h1>
```

```
<CFFORM NAME="Form1" ACTION="CFGRID_2.cfm">
```

```
<CFGRID NAME="TravelerGrid" QUERY="GetTraveler"
```

```
WIDTH="390" HEIGHT="285">
```

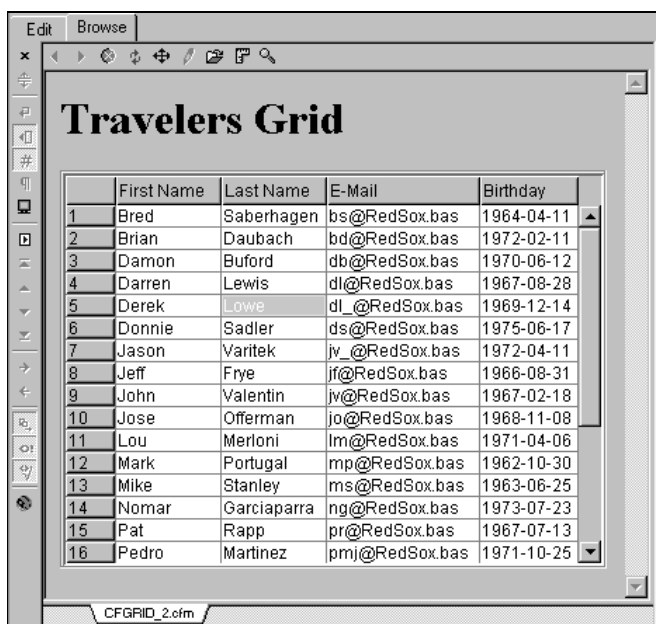
```

AUTOWIDTH="Yes" SELECTMODE="Single">
<CFGRIDCOLUMN NAME="FirstName_tra" HEADER="First Name">
<CFGRIDCOLUMN NAME="LastName_tra" HEADER="Last Name">
<CFGRIDCOLUMN NAME="EMail_tra" HEADER="E-Mail" WIDTH="110">
<CFGRIDCOLUMN NAME="Birthday_tra" HEADER="Birthday" WIDTH="70">
</CFGRID>
</CFFORM>
</BODY>
</HTML>

```

Обращаем ваше внимание на то, что атрибут `HEADER` тега `<CFGRIDCOLUMN>` необязателен, однако полезен, когда текст заголовка колонки требуется специфицировать, как отличный от наименования поля таблицы данных. Атрибут же `NAME` указывает на поле таблицы данных, которые будут отражены в колонке. При отсутствии атрибута `HEADER` значение атрибута `NAME` отражается в качестве заголовка. Используемый в этом примере атрибут `AUTOWIDTH` обеспечивает автоматическое выравнивание ширины каждой колонки так, чтобы все колонки были видимы в пределах ширины, указанной атрибутом `WIDTH` тега `<CFGRID>`. При этом ширина указанных явно колонок остается неизменной.

На рис. 12.10 показан результат приведенного примера.



	First Name	Last Name	E-Mail	Birthday
1	Bred	Saberhagen	bs@RedSox.bas	1964-04-11
2	Brian	Daubach	bd@RedSox.bas	1972-02-11
3	Damon	Buford	db@RedSox.bas	1970-06-12
4	Darren	Lewis	dl@RedSox.bas	1967-08-28
5	Derek	Lowe	dl_@RedSox.bas	1969-12-14
6	Donnie	Sadler	ds@RedSox.bas	1975-06-17
7	Jason	Varitek	jv_@RedSox.bas	1972-04-11
8	Jeff	Frye	jf@RedSox.bas	1966-08-31
9	John	Valentin	jv@RedSox.bas	1967-02-18
10	Jose	Offerman	jo@RedSox.bas	1968-11-08
11	Lou	Merloni	lm@RedSox.bas	1971-04-06
12	Mark	Portugal	mp@RedSox.bas	1962-10-30
13	Mike	Stanley	ms@RedSox.bas	1963-06-25
14	Nomar	Garcia-parra	ng@RedSox.bas	1973-07-23
15	Pat	Rapp	pr@RedSox.bas	1967-07-13
16	Pedro	Martinez	pmj@RedSox.bas	1971-10-25

Рис. 12.10. Пример использования тега `<CFGRID>` с применением запроса

Управляющий элемент <CFGRID> достаточно богат возможностями форматирования фона, данных, заголовка и колонки с порядковыми номерами. В следующем примере тег <CFGRID> дополнен целым рядом атрибутов, отвечающих за визуальное отображение элемента:

```
<HTML>
<HEAD>
  <TITLE>CFGRID</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
    SELECT *
      FROM Traveler
      ORDER BY FirstName_tra, LastName_tra
  </CFQUERY>
  <h1>Travelers Grid</h1>
  <CFFORM NAME="Form1" ACTION="CFGRID_3.cfm">
    <CFGRID NAME="TravelerGrid" QUERY="GetTraveler"
      WIDTH="470" HEIGHT="270" AUTOWIDTH="Yes"
      SELECTMODE="Single"
      COLHEADERALIGN="Center" FONT="Zurich Ex BT" FONTSIZE="11"
      COLHEADERFONT="Zurich BlkEx BT"
      ROWHEADERFONT="Zurich Ex BT"
      COLHEADERFONTSIZE="11" ROWHEADERFONTSIZE="8"
      SELECTCOLOR="##808040" ROWHEIGHT="24"
      BGCOLOR="##EDECE0" TEXTCOLOR="##696934">
      <CFGRIDCOLUMN NAME="FirstName_tra" HEADER="First Name">
      <CFGRIDCOLUMN NAME="LastName_tra" HEADER="Last Name">
      <CFGRIDCOLUMN NAME="Email_tra" HEADER="E-Mail" WIDTH="110">
      <CFGRIDCOLUMN NAME="Birthdays_tra" HEADER="Birthdays" WIDTH="70">
    </CFGRID>
  </CFFORM>
</BODY>
</HTML>
```

Здесь атрибут COLHEADERALIGN позволяет выравнивать заголовок по центру (Center), по левому (Left) или по правому (Right) краю. Атрибутами COLHEADERFONT и ROWHEADERFONT задаем шрифт заголовка колонок и шрифт колонки с порядковыми номерами, атрибутами COLHEADERFONTSIZE и ROWHEADERFONTSIZE назначаем размер шрифта заголовка колонок и колонки с порядковыми номерами. Атрибутом SELECTCOLOR определяем цвет выде-

ленной ячейки или ячеек, если атрибут `SELECTMODE` принимает значение `Row` или `Column`, а также с помощью атрибута `ROWHEIGHT` задаем высоту рядов. В свою очередь атрибутами `FONT`, `FONTSIZE`, `TEXTCOLOR` и `BGCOLOR` назначаем шрифт, размер, цвет и фон текста данных.

На рис. 12.11 показан результат приведенного примера.

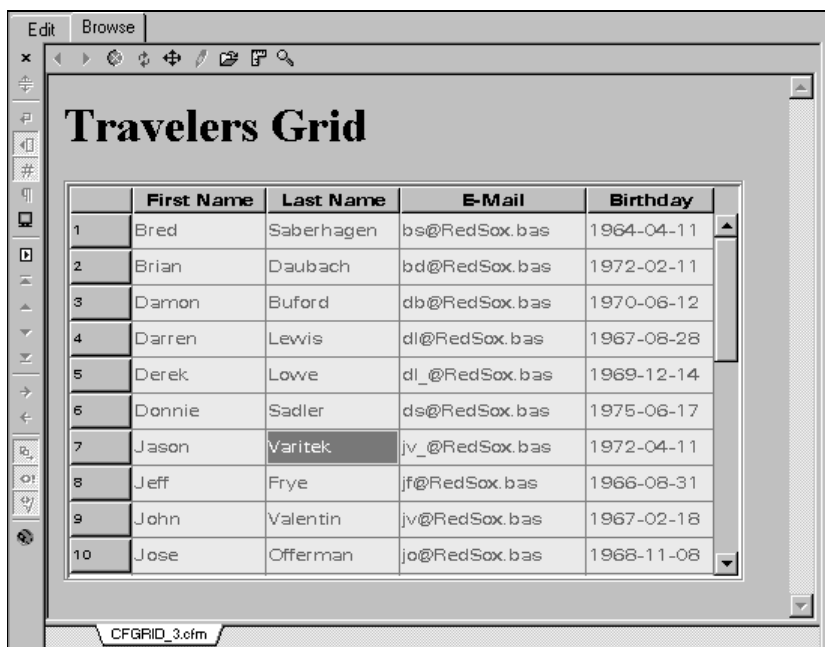


Рис. 12.11. Пример использования тега `<CFGRID>` в собственном стиле

Как мы уже отметили, в качестве типа колонки можно задавать значение, равное `Boolean`, представляя тем самым колонку в виде флажка (checkbox), предназначенного для данных с логическим значением. В следующем примере добавлена новая колонка, отражающая пол зарегистрированных путешественников, в виде флажка:

```
<HTML>
<HEAD>
  <TITLE>CFGRID</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
    SELECT *
    FROM Traveler
```

```

ORDER BY FirstName_tra, LastName_tra
</CFQUERY>
<h1>Travelers Grid</h1>
<CFFORM NAME="Form1" ACTION="CFGRID_4.cfm">
  <CFGRID NAME="TravelerGrid" QUERY="GetTraveler"
    WIDTH="500" HEIGHT="280" AUTOWIDTH="Yes"
    SELECTMODE="Column"
    COLHEADERALIGN="Center" FONT="Zurich Ex BT" FONTSIZE="11"
    COLHEADERFONT="Zurich BlkEx BT"
    ROWHEADERFONT="Zurich Ex BT"
    COLHEADERFONTSIZE="11" ROWHEADERFONTSIZE="8"
    SELECTCOLOR="##808040" ROWHEIGHT="24"
    BGCOLOR="##EDECE0" TEXTCOLOR="##696934"
    SORT="Yes" SORTASCENDINGBUTTON="Asc"
    SORTDESCENDINGBUTTON="Desc">
    <CFGRIDCOLUMN NAME="FirstName_tra" HEADER="First Name">
    <CFGRIDCOLUMN NAME="LastName_tra" HEADER="Last Name">
    <CFGRIDCOLUMN NAME="Email_tra" HEADER="E-Mail" WIDTH="110">
    <CFGRIDCOLUMN NAME="Gender_tra" HEADER="M/W" WIDTH="30"
      TYPE="Boolean">
    <CFGRIDCOLUMN NAME="Birthday_tra" HEADER="Birthday" WIDTH="70">
  </CFGRID>
</CFFORM>
</BODY>
</HTML>

```

В данном фрагменте кода также использован атрибут `SORT` тега `<CFGRID>`, указывающий на размещение в нижней части управляющего элемента дополнительных кнопок для сортировки данных. Атрибуты `SORTASCENDINGBUTTON` и `SORTDESCENDINGBUTTON` позволяют указывать собственный текст на кнопках. По умолчанию в качестве текста кнопок сортировки принимаются значения `SortAsc` и `SortDesc`. Как видно на рис. 12.12, выделена вся текущая колонка, т. к. атрибут `SELECTMODE` принимает значение `Column`.

Тег `<CFGRIDCOLUMN>`, так же, как и тег `<CFTREEITEM>`, поддерживает атрибуты `HREF`, `HREFKEY` и `TARGET`, позволяющие создавать гипертекстовые ссылки, передавать значение выбранной записи в качестве переменной и указывать правило открытия целевой страницы.

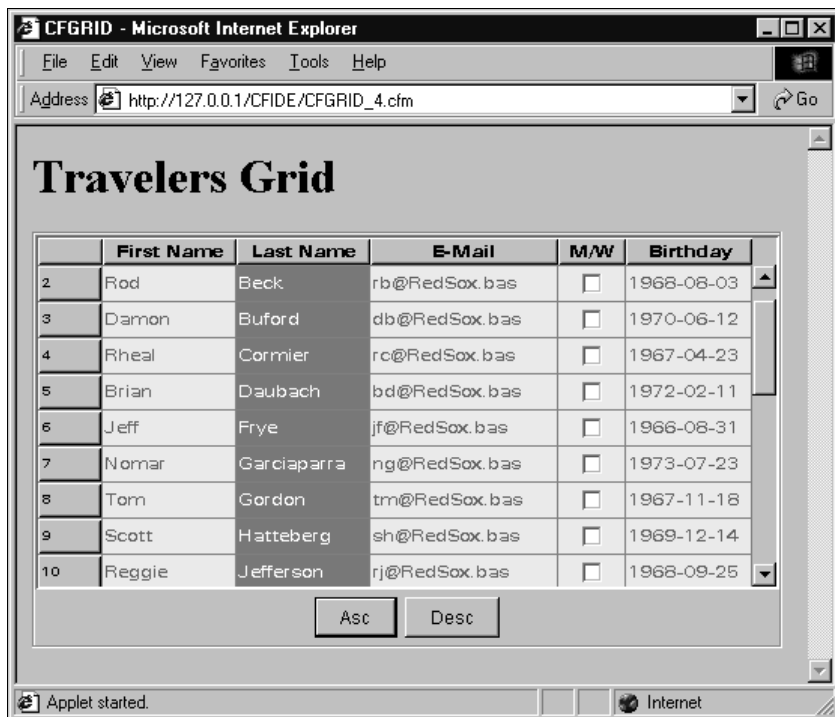


Рис. 12.12. Пример использования тега <CFGRID> с включением флажков и кнопок сортировки данных

Далее приведем пример, где две колонки являются ссылками.

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>CFGRID</TITLE>
```

```
</HEAD>
```

```
<BODY bgcolor="Silver" alink="#107098" link="#0C1C47" vlink="#003366">
```

```
<CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
```

```
SELECT *, 'mailto:' + EMail_tra AS EMail
```

```
FROM Traveler
```

```
ORDER BY FirstName_tra, LastName_tra
```

```
</CFQUERY>
```

```
<h1>Travelers Grid</h1>
```

```
<CFFORM NAME="Form1" ACTION="CFGRID_5.cfm">
```

```
<CFGRID NAME="TravelerGrid" QUERY="GetTraveler"
```

```
WIDTH="500" HEIGHT="280" AUTOWIDTH="Yes"
```

```
SELECTMODE="Single"
```



```

COLHEADERALIGN="Center" FONT="Zurich Ex BT" FONTSIZE="11"
COLHEADERFONT="Zurich BlkEx BT"
ROWHEADERFONT="Zurich Ex BT"
COLHEADERFONTSIZE="11" ROWHEADERFONTSIZE="8"
SELECTCOLOR="##808040" ROWHEIGHT="24"
BGCOLOR="##EDECE0" TEXTCOLOR="##696934"
SORT="Yes" PICTUREBAR="Yes">
<CFGRIDCOLUMN NAME="Key_tra" DISPLAY="No">
<CFGRIDCOLUMN NAME="FirstName_tra" HEADER="First Name"
    HREF="DetalTraveler.cfm" HREFKEY="Key_tra">
<CFGRIDCOLUMN NAME="LastName_tra" HEADER="Last Name">
<CFGRIDCOLUMN NAME="Email" DISPLAY="No">
<CFGRIDCOLUMN NAME="Email_tra" HEADER="E-Mail" WIDTH="110"
    HREF="Email">
<CFGRIDCOLUMN NAME="Gender_tra" HEADER="M/W" WIDTH="30"
    TYPE="Boolean">
<CFGRIDCOLUMN NAME="Birthday_tra" HEADER="Birthday" WIDTH="70">
</CFGRID>
</CFFORM>
</BODY>
</HTML>

```

Правило передачи значений выбранной записи при использовании гипертекстовых ссылок тега `<CFGRIDCOLUMN>` аналогично правилу передачи значений тега `<CFTREEITEM>`. В этом же примере мы использовали атрибут `PICTUREBAR`, предназначенный для вывода на кнопках вспомогательных изображений, характеризующих возможные действия.

На рис. 12.13 показан результат приведенного примера.

Предыдущая глава была полностью посвящена модификации, добавлению и удалению данных. Так вот, управляющий элемент `<CFGRID>` позволяет облегчить процесс формирования интерфейса для решения подобных задач. Заданием значения атрибута `SELECTMODE` тега `<CFGRID>` равным `Edit` предполагается возможность редактирования данных непосредственно внутри построенной таблицы данных. Указав же при этом атрибуты `INSERT` и `DELETE`, как показано в следующем примере, в нижней части управляющего элемента появляются дополнительные кнопки, позволяющие выполнять соответствующие действия (вставить и удалить). Впоследствии же остается только обработать произведенные изменения в таблице и применить эти изменения к базе данных, являющейся источником управляющего элемента `<CFGRID>`.

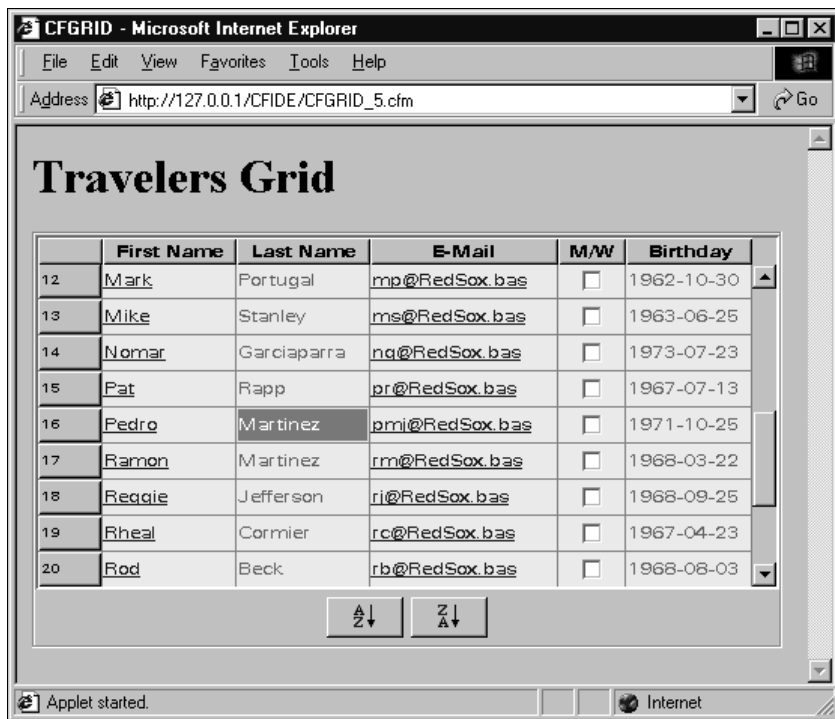


Рис. 12.13. Пример использования тега <CFGRID> с применением гипертекстовых ссылок

При редактировании данных, используя управляющий элемент <CFGRID>, ColdFusion создает следующие массивы:

- ❑ `GridName.ColName[RecordIndex]` — хранит новое значение отредактированной ячейки;
- ❑ `GridName.Original.ColName[RecordIndex]` — хранит первоначальное значение отредактированной ячейки;
- ❑ `GridName.RowStatus.Action[RecordIndex]` — хранит тип изменений отредактированной записи таблицы: D — для удаления, I — для вставки или U — для обновления.

Например, для обработки выполненных изменений при передаче данных на страницу, указанную атрибутом ACTION тега <CFFORM>, можно сформировать следующий код:

```
<CFIF isDefined("Form.TravelerGrid.RowStatus.Action")>
  <CFLOOP INDEX="Record" FROM="1"
    TO=#ArrayLen(Form.TravelerGrid.RowStatus.Action)>#
    <CFSWITCH EXPRESSION=#Form.TravelerGrid.RowStatus.Action[Record]>#
```

```

<CFCASE VALUE="U">
    <!--Обновление (UPDATE)--->
</CFCASE>
<CFCASE VALUE="I">
    <!--Вставка (INSERT)--->
</CFCASE>
<CFCASE VALUE="D">
    <!--Удаление (DELETE)--->
</CFCASE>
</CFSWITCH>
</CFLOOP>
</CFIF>

```

где TravelerGrid — имя элемента <CFGRID>.

Следующий пример состоит из двух частей или вернее из двух страниц: CFGRID_6.cfm (листинг 12.1), где расположена таблица, и RecordTravelers.cfm (листинг 12.2), где происходит обработка и принятие исправлений. Далее происходит возвращение на страницу с таблицей.

Листинг 12.1. Код страницы CFGRID_6.cfm

```

<HTML>
<HEAD>
    <TITLE>CFGRID</TITLE>
</HEAD>
<BODY bgcolor="Silver">
    <CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
        SELECT *
        FROM Traveler
        ORDER BY FirstName_tra, LastName_tra
    </CFQUERY>
    <h1>Travelers Grid</h1>
    <CFFORM NAME="Form1" ACTION="RecordTravelers.cfm">
        <CFGRID NAME="TravelerGrid" QUERY="GetTraveler"
            WIDTH="500" HEIGHT="280" AUTOWIDTH="Yes" SELECTMODE="Edit"
            COLHEADERALIGN="Center" FONT="Zurich Ex BT" FONTSIZE="11"
            COLHEADERFONT="Zurich BlkEx BT"
            ROWHEADERFONT="Zurich Ex BT"
            COLHEADERFONTSIZE="11" ROWHEADERFONTSIZE="8"
            SELECTCOLOR="##808040" ROWHEIGHT="24"
            BGCOLOR="##EDEC0" TEXTCOLOR="##696934"

```

```

        SORT="Yes" INSERT="Yes" DELETE="Yes" PICTUREBAR="Yes">
<CFGRIDCOLUMN NAME="Key_tra" DISPLAY="No">
<CFGRIDCOLUMN NAME="FirstName_tra" HEADER="First Name">
<CFGRIDCOLUMN NAME="LastName_tra" HEADER="Last Name">
<CFGRIDCOLUMN NAME="Email_tra" HEADER="E-Mail" WIDTH="110">
<CFGRIDCOLUMN NAME="Gender_tra" HEADER="M/W" WIDTH="30"
        TYPE="Boolean">
<CFGRIDCOLUMN NAME="Birthday_tra" HEADER="Birthday" WIDTH="70"
        BGCOLOR="(CX GT 1970-06-06 ? EDECE0 : Orange)">
</CFGRID>
<br><input type='Submit' value='Record'>
</CFFORM>
</BODY>
</HTML>

```

На рис. 12.14 приведен вид страницы CFGRID_6.cfm.



Рис. 12.14. Пример использования тега <CFGRID> с применением режима редактирования

Листинг 12.2. Код страницы RecordTravelers.cfm

```

<HTML>
<HEAD>
  <TITLE>CFGRID</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFIF isDefined("Form.TravelerGrid.RowStatus.Action")>
    <CFLOOP INDEX="Record" FROM="1"
      TO=#ArrayLen(Form.TravelerGrid.RowStatus.Action)#>
    <CFSWITCH EXPRESSION=#Form.TravelerGrid.RowStatus.Action[Record]#>
    <CFCASE VALUE="U">
      <CFQUERY NAME="UpdateTraveler" DATASOURCE="#application.dsn#">
        UPDATE Traveler
        SET FirstName_tra='#Form.TravelerGrid.FirstName_tra[Record]#',
          LastName_tra='#Form.TravelerGrid.LastName_tra[Record]#',
          Email_tra='#Form.TravelerGrid.Email_tra[Record]#',
          Gender_tra='#Form.TravelerGrid.Gender_tra[Record]#',
          Birthday_tra='#Form.TravelerGrid.Birthday_tra[Record]#'
        WHERE Key_tra=#Form.TravelerGrid.Original.Key_tra[Record]#
      </CFQUERY>
    </CFCASE>
    <CFCASE VALUE="I">
      <CFQUERY NAME="InsertTraveler" DATASOURCE="#application.dsn#">
        INSERT INTO Traveler(FirstName_tra, LastName_tra, Email_tra,
          Gender_tra, Birthday_tra, Password_tra)
        VALUES ('#Form.TravelerGrid.FirstName_tra[Record]#',
          '#Form.TravelerGrid.LastName_tra[Record]#',
          '#Form.TravelerGrid.Email_tra[Record]#',
          '#Form.TravelerGrid.Gender_tra[Record]#',
          '#Form.TravelerGrid.Birthday_tra[Record]#', '')
      </CFQUERY>
    </CFCASE>
    <CFCASE VALUE="D">
      <CFQUERY NAME="DeleteTraveler" DATASOURCE="#application.dsn#">
        DELETE FROM Traveler
        WHERE Key_tra=#Form.TravelerGrid.Original.Key_tra[Record]#
      </CFQUERY>
    </CFCASE>
  </CFLOOP>
</BODY>
</HTML>

```

```

</CFSWITCH>
</CFLOOP>
</CFIF>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=CFGRID_6.cfm">
</BODY>
</HTML>

```

На странице CFGRID_6.cfm для поля с днями рождения был применен атрибут BGCOLOR со встроенным выражением. Все дело в том, что для атрибутов TEXTCOLOR и BGCOLOR ColdFusion дает возможность вместо конкретного значения использовать выражение, позволяющее определять значение по условию. Причем формат выражения должен выглядеть следующим образом:

```
(CX Operator String ? TrueCondition : FalseCondition)
```

где:

- ❑ CX — колонка, по значению которой проверяется условие. Для текущей колонки следует использовать CX, а для конкретной колонки указывать Cn, где n — порядковый номер колонки;
- ❑ Operator — в качестве оператора можно использовать: EQ (равно), GT (больше) или LT (меньше);
- ❑ String — строка, начиная с которой проверяется значение указанной колонки;
- ❑ TrueCondition — значение атрибута при соблюдении условия;
- ❑ FalseCondition — значение атрибута при несоблюдении условия.

В нашем примере конструкция выглядит так:

```
BGCOLOR="(CX GT 1970-06-06 ? EDECE0 : Orange)"
```

Что позволяет выделить цветом EDECE0 сведения о тех, кто родился до 6 июня 1970 года, и закрасить оранжевым цветом даты рождения всех остальных.

Тег <CFGRIDCOLUMN> позволяет определять маску (mask) колонки, используя атрибут NUMBERFORMAT подобно функции ColdFusion NumberFormat. Применяется для числовых значений, например:

```
<CFGRIDCOLUMN NAME="CostAdult_tou" HEADER="Adults" NUMBERFORMAT="$__."__">
```

Тег <CFGRID>, так же, как теги <CFINPUT>, <CFTEXTINPUT>, <CFSLIDER> и <CFTREE>, поддерживает атрибуты ONERROR и ONVALIDATE.

Тег <CFAPPLET>

Тег <CFAPPLET> предназначен для включения в код собственных Java-апплетов. Используемый Java-апплет предварительно должен быть зарегистрирован на странице администратора "ColdFusion Administrator" в разделе **Server | Extensions | Java Applets** (Сервер | Расширения | Java-апплеты).

На рис. 12.15 продемонстрирована форма регистрации Java-апплета COPYTEXT, входящего в поставку программного пакета ColdFusion Server.

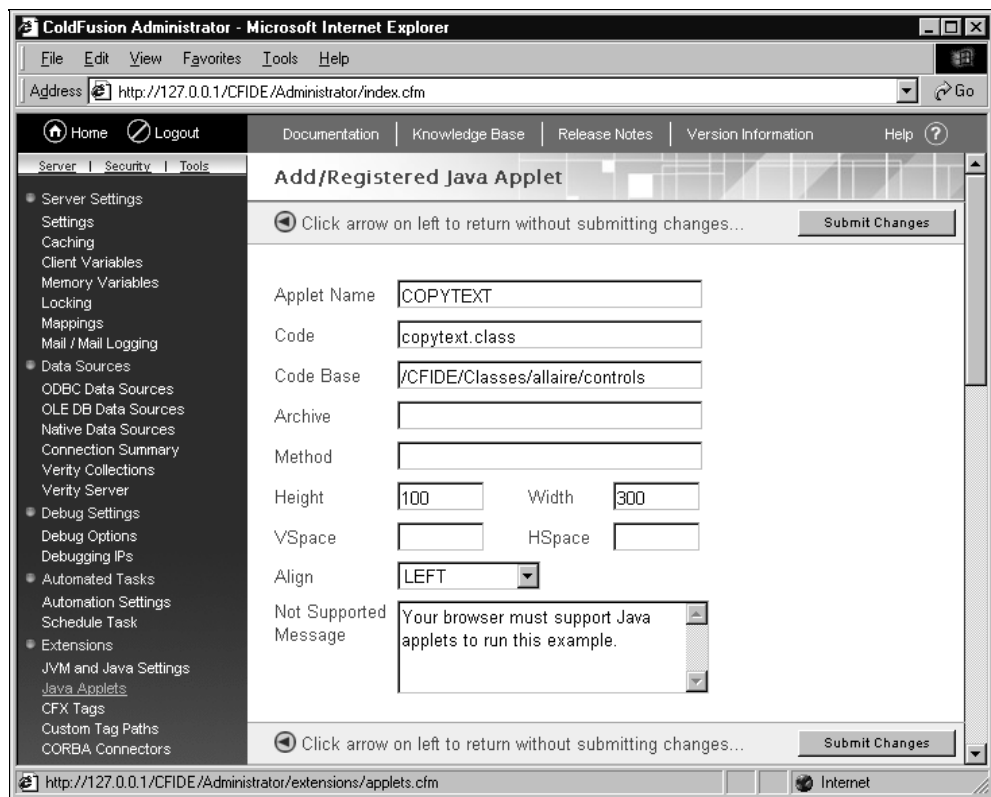


Рис. 12.15. Страница "ColdFusion Administrator", раздел **Java Applets**

Здесь:

- ❑ **Applet Name** (Имя апплета) — наименование апплета;
- ❑ **Code** (Код) — имя файла, содержащего апплет;
- ❑ **Code Base** (Основной код) — путь к каталогу, содержащему файл апплета; данное значение нет необходимости указывать при использовании значения **Archive** (Архив);
- ❑ **Archive** (Архив) — указание на архивный jar-файл, содержащий апплет;
- ❑ **Method** (Метод) — наименование метода, который возвращает символьное значение. Наименование метода используется с тегом `<CFAPPLET>` в качестве атрибута. Если апплет не имеет метода, этот пункт не указывается;

- ❑ **Height** (Высота) — высота области апплета в пикселах; соответствует атрибуту `HEIGHT` тега `<CFAPPLET>`;
- ❑ **Width** (Ширина) — ширина области апплета в пикселах; соответствует атрибуту `WIDTH` тега `<CFAPPLET>`;
- ❑ **VSpace** (Вертикальный интервал) — величина незаполненного пространства сверху и снизу от апплета в пикселах; соответствует атрибуту `VSPACE` тега `<CFAPPLET>`;
- ❑ **HSpace** (Горизонтальный интервал) — величина незаполненного пространства слева и справа от апплета в пикселах; соответствует атрибуту `HSPACE` тега `<CFAPPLET>`;
- ❑ **Align** (Выводить) — выравнивание апплета. (**LEFT**, **RIGHT**, **BOTTOM**, **TOP**, **TEXTTOP**, **MIDDLE**, **ABSMIDDLE**, **BASELINE** или **ABSBOTTOM**); соответствует атрибуту `ALIGN` тега `<CFAPPLET>`;
- ❑ **Not Supported Message** (Сообщение при отсутствии поддержки) — текст сообщения, выводимого на экран, при отсутствии поддержки браузером Java-апплетов; соответствует атрибуту `NOTSUPPORTED` тега `<CFAPPLET>`;
- ❑ **Parameter Name** (Имя параметра) — перечень наименований необходимых параметров при использовании апплета;
- ❑ **Value** (Значение) — перечень значений указанных параметров.

Также необходимые параметры можно указывать с помощью тега `<CFAPPLET>` при непосредственном использовании Java-апплета.

Приведенным ниже примером мы завершаем описание построения форм на основе тега `<CFFORM>`, где интегрируем в CFML-страницу Java-апплет `Copytext`:

```
<HTML>
<HEAD>
  <TITLE>CFAPPLET</TITLE>
</HEAD>
<BODY bgcolor="Silver">
  <CFFORM NAME="Form1" ACTION="CFAPPLET_.cfm">
    <CFAPPLET APPLETSOURCE="Copytext" NAME="Copytext"
      WIDTH="250" HEIGHT="70">
  </CFFORM>
</BODY>
</HTML>
```

При использовании тега `<CFAPPLET>` обязательными атрибутами являются:

- ❑ `APPLETSOURCE` — наименование зарегистрированного Java-апплета;
- ❑ `NAME` — имя переменной формы для Java-апплета.

Отметим, что все управляющие элементы тега формы `<CFFORM>`, основанные на Java-апплетах, а это `<CFTEXTINPUT>`, `<CFSLIDER>`, `<CFTREE>`, `<CFGRID>` и `<CFAPPLET>`, поддерживают атрибут `NOTSUPPORTED`, позволяющий указывать текст сообщения, выводимого на экран, при отсутствии поддержки браузером Java-апплетов.

Резюме

Подведем итог. В данной главе мы использовали CFML-теги: `<CFFORM>`, `<CFINPUT>`, `<CFSELECT>`, `<CFTEXTINPUT>`, `<CFSLIDER>`, `<CFTREE>`, `<CFTREEITEM>`, `<CFGRID>`, `<CFGRIDCOLUMN>`, `<CFGRIDROW>`, `<CFAPPLET>`.

Глава 13



Массивы, структуры и списки CFML

Как было сказано в *главе 9*, ColdFusion доступно достаточно большое количество типов переменных. В той же главе мы постарались описать каждый из существующих типов, к которым относятся и такие типы, как *массивы* (arrays), *структуры* (structures) и *списки* (lists). На наш взгляд эти типы заслуживают отдельного внимания. При большом количестве отдельных переменных более эффективно использовать вместо них массивы, структуры или списки.

Массивы

Массивы (arrays) — это группа переменных одного типа, доступ к которым осуществляется с помощью общего имени. Хотя функция `ArrayNew` только поддерживает создание массивов до трехмерной размерности, массив не имеет предела по размеру¹ или максимальное значение².

Надо отметить, что массивы ColdFusion несколько отличаются от традиционных массивов, т. к. они являются динамическими, и если для обычных массивов их размер постоянен и симметричен, то массивы ColdFusion могут иметь ряды, отличающиеся друг от друга в зависимости от добавленных или удаленных данных.

Создание массива

Для использования массива необходимо предварительно его объявить:

```
<CFSET MyNewArray = ArrayNew(x)>
```

где *x* — число, отражающее размерность массива, которое может принимать значение от 1 до 3.

¹ То есть количество элементов. — *Ред.*

² Максимальное значение ограничивается лишь возможностями вашего компьютера. — *Ред.*

К примеру, создадим одномерный массив и запишем в уже существующий массив два элемента следующим образом:

```
<CFSET CruiseArray = ArrayNew(1)>
<CFSET CruiseArray[1] = "Museum of Jewish Horitage">
<CFSET CruiseArray[2] = "Ellis Island">
```

В следующем примере создадим двумерный массив и заполним его:

```
<CFSET TravelerArray = ArrayNew(2)>
<CFSET TravelerArray[1][1] = "Nomar">
<CFSET TravelerArray[1][2] = "GarciaParra">
<CFSET TravelerArray[2][1] = "Pedro">
<CFSET TravelerArray[2][2] = "Martinez">
```

Как уже было замечено, максимальное значение размерности массива в ColdFusion определяется значением 3 (трехмерный). Однако всегда можно создать массив массивов, т. е. когда массиву назначается вложенный элемент, являющийся также массивом. При этом за счет вложенного массива увеличивается и размерность внешнего массива. Например:

```
<CFSET TravelerArray = ArrayNew(2)>
<CFSET TicketArray = ArrayNew(3)>
<CFSET TicketArray[3][2][1] = TravelerArray>
<CFSET TicketArray[3][2][1][1][1] = "Buying">
```

Проверка массива

- ❑ Функция `ArrayIsEmpty(array)` проверяет, является ли массив пустым.
- ❑ Функция `ArrayLen(array)` возвращает длину заданного массива.
- ❑ Функция `IsArray(value [, number])` возвращает `True`, если значение является массивом. Здесь `number` — число, позволяющее уточнить проверку соответствия количества элементов массива.

Добавление элементов

- ❑ Функция `ArrayAppend(array, value)` добавляет элемент в конец заданного массива.
- ❑ Функция `ArrayPrepend(array, value)` добавляет элемент в начало заданного массива.
- ❑ Функция `ArrayInsertAt(array, position, value)` вставляет элемент массива на указанную позицию.
- ❑ Функция `ArraySet(array, start_pos, end_pos, value)` присваивает определенное значение ряду элементов одномерного массива в указанном диапазоне.

Например:

```
<HTML>
  <HEAD>
    <TITLE>myArray</TITLE>
  </HEAD>
  <BODY>
    <CFSET CruiseArray = ArrayNew(1)>
    <CFSET CruiseArray[1] = "World Financial Center">
    <CFSET CruiseArray[2] = "Ellis Island">
    <CFSET ArrayPrepend(CruiseArray, "World Trade Center")>
    <CFSET ArrayAppend(CruiseArray, "Statue of Liberty")>
    <CFSET ArrayInsertAt(CruiseArray, 3, "Battery Park")>
    <CFSET ArraySet(CruiseArray, 6, 8, "Repair")>

    <CFIF IsArray(CruiseArray) AND ArrayIsEmpty(CruiseArray) eq false>
      <CFLOOP INDEX = "1" FROM = "1" TO = #ArrayLen(CruiseArray)#>
        <CFOUTPUT>#i#. #CruiseArray[i]#</CFOUTPUT><br>
      </CFLOOP>
    </CFIF>
  </BODY>
</HTML>
```

В результате приведенного выше примера создается одномерный массив `CruiseArray` с восемью элементами. При этом значения элементов массива выводятся на экран в следующем виде:

1. World Trade Center
2. World Financial Center
3. Battery Park
4. Ellis Island
5. Statue of Liberty
6. Repair
7. Repair
8. Repair

Также заполнение массива можно осуществлять непосредственно значениями из базы данных, используя для этой цели тег условного цикла `<CFLOOP>` с атрибутом `QUERY`, как показано ниже:

```
<HTML>
  <HEAD>
    <TITLE>myArray2d</TITLE>
```

```

</HEAD>
<BODY>
  <CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
    SELECT Key_tou, Name_tou
      FROM Tour
     ORDER BY Key_tou
  </CFQUERY>
  <CFSET TourArray2d = ArrayNew(2)>
  <CFLOOP QUERY="GetTour">
    <CFSET TourArray2d[CurrentRow][1] = Key_tou[CurrentRow]>
    <CFSET TourArray2d[CurrentRow][2] = Name_tou[CurrentRow]>
  </CFLOOP>
  <CFIF IsArray(TourArray2d) AND ArrayIsEmpty(TourArray2d) eq false>
    <CFLOOP INDEX = "i" FROM = "1" TO = #ArrayLen(TourArray2d)#>
      <CFOUTPUT>#TourArray2d[i][1]# #TourArray2d[i][2]#</CFOUTPUT><br>
    </CFLOOP>
  </CFIF>
</BODY>
</HTML>

```

В результате выполнения приведенного примера создается двумерный массив `TourArray2d` с присвоением его элементам значений в зависимости от содержимого базы данных и последующим выводом их на экран.

Регулирование положением элементов массива

❑ Функция `ArraySort(array, sort_type [, sort_order])` сортирует элементы массива.

Для функции `ArraySort` можно задавать следующие типы сортировки (`sort_type`):

- `Numeric` — сортировка числовых величин;
- `Text` — сортировка в алфавитном порядке, сначала выводятся значения с учетом верхнего регистра, затем нижнего;
- `TextNoCase` — сортировка в алфавитном порядке без учета чувствительности к регистру.

Также функция `ArraySort` поддерживает сортировку (`sort_order`):

- `Asc` — по возрастанию (по умолчанию);
- `Desc` — по убыванию.

Например, для сортировки уже существующего массива `CruiseArray` в алфавитном порядке по возрастанию достаточно включить следующую строку кода:

```
<CFSET ArraySort(CruiseArray, "Text", "Asc")>
```

В результате элементы для нашего примера будут отсортированы следующим образом:

1. Battery Park
2. Ellis Island
3. Repair
4. Repair
5. Repair
6. Statue of Liberty
7. World Financial Center
8. World Trade Center

- ❑ Функция `ArraySwap(array, position1, position2)` переставляет местами элементы массива.

Например, чтобы переставить местами первый и шестой элементы в уже отсортированном массиве `CruiseArray`, достаточно включить следующую строку кода:

```
<CFSET ArraySwap(CruiseArray, 1, 6)>
```

В результате список элементов массива `CruiseArray` приобретет следующий вид:

1. Statue of Liberty
2. Ellis Island
3. Repair
4. Repair
5. Repair
6. Battery Park
7. World Financial Center
8. World Trade Center

Удаление элементов

- ❑ Функция `ArrayClear(array)` удаляет все данные из массива.

Например, для очистки уже существующего массива `CruiseArray` достаточно включить следующую строку кода:

```
<CFSET ArrayClear(CruiseArray)>
```

- ❑ Функция `ArrayDeleteAt(array, position)` удаляет элемент массива, находящийся на указанной позиции.

Если есть необходимость удалить третий по счету элемент массива `CruiseArray`, следует включить строку кода, показанную ниже:

```
<CFSET ArrayDeleteAt(CruiseArray, 3)>
```

Обработка значений элементов

- ❑ Функция `ArrayAvg(array)` возвращает среднее значение по всем элементам массива.
- ❑ Функция `ArrayMax(array)` возвращает максимальное значение элементов массива.
- ❑ Функция `ArrayMin(array)` возвращает минимальное значение элементов массива.
- ❑ Функция `ArraySum(array)` возвращает сумму значений элементов массива.

Например:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>myArray</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn">
```

```
    SELECT CostAdult_tou
```

```
    FROM Tour
```

```
</CFQUERY>
```

```
<CFSET CostArray = ArrayNew(1)>
```

```
<CFLOOP QUERY="GetTour">
```

```
    <CFSET CostArray[CurrentRow] = CostAdult_tou[CurrentRow]>
```

```
</CFLOOP>
```

```
<CFIF IsArray(CostArray) AND ArrayIsEmpty(CostArray) eq false>
```

```
    <CFLOOP INDEX = "1" FROM = "1" TO = #ArrayLen(CostArray)#>
```

```
        <CFOUTPUT>#CostArray[i]#</CFOUTPUT><br>
```

```
    </CFLOOP>
```

```
</CFIF>
```

```
<br>
```

```
<CFOUTPUT>CostAvg: #ArrayAvg(CostArray)#</CFOUTPUT><br>
```

```
<CFOUTPUT>CostMax: #ArrayMax(CostArray)#</CFOUTPUT><br>
```

```
<CFOUTPUT>CostMin: #ArrayMin(CostArray)#</CFOUTPUT><br>
```

```
<CFOUTPUT>CostSum: #ArraySum(CostArray)#</CFOUTPUT><br>
</BODY>
</HTML>
```

Результат работы приведенного выше кода таков:

```
13.00
20.00
24.00
20.00

CostAvg: 19.25
CostMax: 24
CostMin: 13
CostSum: 77
```

Преобразование массивов

Функция `ArrayToList(array [, delimiter])` возвращает список элементов, преобразуя данные из массива. Здесь `delimiter` воспринимается как разделитель. По умолчанию разделителем является запятая (,).

Например, для создания списка, получаемого преобразованием данных уже существующего массива `CostArray`, достаточно включить следующий код:

```
<CFSET CostList = ArrayToList(CostArray)>
<CFOUTPUT>#CostList#</CFOUTPUT>
```

Результатом будет строка, отображающая содержание нового списка:

```
13.00,20.00,24.00,20.00
```

Или, к примеру:

```
<CFSET CostList2 = ArrayToList(CostArray, "---")>
<CFOUTPUT>#CostList2#</CFOUTPUT>
```

Здесь же результат несколько изменится и будет таким:

```
13.00---20.00---24.00---20.00
```

Размер массива

Функция `ArrayResize(array, minimum_size)` задает размер массива.

Например:

```
<CFSET ArrayResize(CruiseArray, 30)>
```

Таким образом, мы определяем минимальный размер массива, равный 30 элементам.

Структуры

Вы можете использовать *структуры* (structures), чтобы создавать и поддерживать пары ключевых значений, обращаться к связанным символьным значениям как к общности, а не как к индивидуальным значениям, или создавать ассоциативные массивы.

Создание структуры

Для использования структуры необходимо предварительно ее объявить:

```
<CFSET MyNewStruct = StructNew()>
```

К примеру, создадим структуру и присвоим значения двум ее элементам:

```
<CFSET TravelerStruct = StructNew()>
```

```
<CFSET TravelerStruct["RS1_98"] = "Nomar Garciaparra">
```

```
<CFSET TravelerStruct["RS2_98"] = "Pedro Martinez">
```

При этом был использован синтаксис ассоциативных массивов. Однако подобного результата можно добиться, применяя объектно-ориентированный синтаксис доступа к элементам, например:

```
<CFSET TravelerStruct = StructNew()>
```

```
<CFSET TravelerStruct.RS1_98 = "Nomar Garciaparra">
```

```
<CFSET TravelerStruct.RS2_98 = "Pedro Martinez">
```

Впрочем, в случае объектно-ориентированного синтаксиса необходимо, чтобы ключ, указанный после точки, начинался с буквы, и в названии ключа были использованы буквы, цифры и символы подчеркивания без разделительных пробелов. Приведем пример воспринимаемых и невоспринимаемых наименований ключей при использовании объектно-ориентированного синтаксиса.

Воспринимаемые наименования:

☐ RS1_98

☐ R1_S_98

☐ R12345

Невоспринимаемые наименования:

☐ _RS1_98

☐ RS1-98

☐ RS1:98

☐ 9RS1_98

Также присвоение значений элементов структуры можно осуществлять с помощью функций.

Проверка структуры

- ❑ Функция `StructIsEmpty(structure)` проверяет, является ли структура пустой.
- ❑ Функция `StructCount(structure)` возвращает количество ключей в заданной структуре.
- ❑ Функция `StructKeyExists(structure, key)` проверяет наличие ключа в массиве.
- ❑ Функция `IsStruct(variable)` возвращает `True`, если значение является структурой.

Добавление элементов и модификация структуры

- ❑ Функция `StructInsert(structure, key, value [, allowoverwrite])` вставляет пару "ключ/значение" в заданную структуру. Здесь `allowoverwrite` указывает на возможность переписывать ключи (по умолчанию `False`).
- ❑ Функция `StructUpdate(structure, key, value)` производит обновление данных в структуре.
- ❑ Функция `StructAppend(struct1, struct2, overwriteFlag)` добавляет в конец одной структуры другую структуру. Здесь `struct1` — структура, в конец которой происходит добавление данных из структуры `struct2`. Если `overwriteFlag` равно `Yes` (по умолчанию), то значения структуры `struct2` переписывают соответствующие значения в структуре `struct1`.

Например:

```
<HTML>
<HEAD>
  <TITLE>myStruct</TITLE>
</HEAD>
<BODY>
  <CFSET TravelerStruct = StructNew()>
  <CFSET StructInsert(TravelerStruct, "RS1_98", "Nomar Garciaparra")>
  <CFSET StructInsert(TravelerStruct, "RS2_98", "Pedro Martinez")>
  <CFSET StructInsert(TravelerStruct, "RS3_98", "Tom Gordon")>
  <CFSET StructInsert(TravelerStruct, "RS4_98", "Mike Stanley")>
  <CFSET StructInsert(TravelerStruct, "RS5_98", "Reggie Jefferson")>
  <CFSET StructInsert(TravelerStruct, "RS6_98", "Mark Portugal")>

  <CFSET TravelerStruct2 = StructNew()>
  <CFSET StructInsert(TravelerStruct2, "RS7_98", "John Valentin")>
```

```

<CFSET StructInsert(TravelerStruct2, "RS8_98", "Jose Offerman")>
<CFSET StructInsert(TravelerStruct2, "RS9_98", "Jason Varitek")>

<CFSET StructUpdate(TravelerStruct2, "RS9_98", "Trot Nixon")>
<CFSET StructAppend(TravelerStruct, TravelerStruct2)>

<CFIF IsStruct(TravelerStruct) AND StructIsEmpty(TravelerStruct)
    eq false>
    <CFLOOP COLLECTION = #TravelerStruct# ITEM = "i">
        <CFOUTPUT>#i# - #TravelerStruct[i]#</CFOUTPUT><br>
    </CFLOOP>
</CFIF>
</BODY>
</HTML>

```

В результате выполнения приведенного выше примера создается структура `TravelerStruct` с девятью значениями в соответствии с ключами. При этом значения структуры выводятся на экран в порядке добавления:

```

RS1_98 - Nomar Garciaparra
RS2_98 - Pedro Martinez
RS3_98 - Tom Gordon
RS4_98 - Mike Stanley
RS5_98 - Reggie Jefferson
RS6_98 - Mark Portugal
RS7_98 - John Valentin
RS8_98 - Jose Offerman
RS9_98 - Trot Nixon

```

Удаление элементов структуры

- ❑ Функция `StructClear(structure)` удаляет все данные из структуры.

Например, для очистки уже существующей структуры `TravelerStruct` достаточно включить следующую строку кода:

```
<CFSET StructClear(TravelerStruct)>
```

- ❑ Функция `StructDelete(structure, key [, indicatenotexisting])` удаляет элемент структуры с указанным значением ключа. Здесь `indicatenotexisting` указывает на необходимость возвращения отрицательного значения при попытке удаления элемента с несуществующим ключом. По умолчанию `No`.

Если есть необходимость удалить элемент структуры `TravelerStruct` со значением ключа `RS7_98` и при этом убедиться, что попытка удаления оказалась удачной, необходимо включить код, показанный ниже:

```
<CFSET isDel = StructDelete(TravelerStruct, "RS7_98","Yes")>
<CFIF isDel eq true>
    Item Delete
</CFIF>
```

Или так:

```
<CFIF StructKeyExists(TravelerStruct, "RS7_98")>
    <CFSET StructDelete(TravelerStruct, "RS7_98")>
</CFIF>
```

Преобразование структур

- ❑ Функция `StructKeyArray(structure)` возвращает массив ключей, преобразуя данные из структуры.
- ❑ Функция `StructKeyList(structure, [delimiter])` возвращает список ключей, преобразуя данные из структуры.

Например, для создания списка ключей, получаемого преобразованием данных уже существующей структуры `TravelerStruct`, достаточно включить следующий код:

```
<CFSET TravelerList = StructKeyList(TravelerStruct)>
<CFOUTPUT>#TravelerList#</CFOUTPUT>
```

Результатом чего будет строка, отображающая содержание нового списка:

```
RS1_98,RS2_98,RS3_98,RS4_98,RS5_98,RS6_98,RS7_98,RS8_98,RS9_98
```

Другие функции обработки структур

- ❑ Функция `StructCopy(structure)` копирует структуру.

Например, следующая строка позволяет скопировать структуру `TravelerStruct` во вновь созданную структуру `TravCopyStruct`:

```
<CFSET TravCopyStruct = StructCopy(TravelerStruct)>
```

- ❑ Функция `StructFind(structure, key)` выполняет поиск данных по ключу в указанной структуре.

Например, для осуществления поиска значений по ключу `RS4_98` в структуре `TravelerStruct` необходимо включить следующую пару строк кода:

```
<CFSET ResultFind = StructFind(TravelerStruct, "RS4_98")>
<CFOUTPUT>#ResultFind#</CFOUTPUT>
```

Результатом для нашего примера будет вывод найденного значения:

Mike Stanley

Списки

Списки (lists) — специальный символьный тип, составленный из элементов, отделенных разделителями.

Вы можете самостоятельно определять разделитель для списка. Причем разделитель может состоять из нескольких символов. По умолчанию в качестве разделителя принимается запятая (,).

Разделители перед первым элементом и после последнего элемента игнорируются.

Структура списков — плоская, т. е. списки не могут быть вложены друг в друга. Также, списки не могут содержать никакие "пустые" элементы. Однако список может быть пуст. При этом он эквивалентен пустому значению "".

В следующих разделах при описании функций обработки списков параметр `delimiter` необходимо воспринимать как разделитель.

Поиск, сортировка, проверка и просмотр списков

- ❑ Функция `ListFind(list, value [, delimiters])` осуществляет поиск значений в списке с заданным разделителем.
- ❑ Функция `ListFindNoCase(list, value [, delimiters])` выполняет поиск значений в списке без учета регистра.
- ❑ Функция `ListContains (list, substring [, delimiters])` возвращает позицию в списке, начиная с которой встречается заданный фрагмент текста (подстрока).
- ❑ Функция `ListContainsNoCase (list, substring [, delimiters])` возвращает позицию в списке, начиная с которой встречается заданный фрагмент текста, без учета регистра.
- ❑ Функция `ListSort(list, sort_type [, sort_order] [, delimiter])` сортирует элементы списка с учетом разделителя.

Для функции `ListSort` так же, как и для функции `ArraySort`, можно задавать следующие типы сортировки (`sort_type`):

- `Numeric` — сортировка числовых величин;
- `Text` — сортировка в алфавитном порядке, сначала выводятся значения с учетом верхнего регистра, затем нижнего;
- `TextNoCase` — сортировка в алфавитном порядке без учета чувствительности к регистру.

Также функции `ListSort` поддерживает сортировку (`sort_order`):

- `Asc` — по возрастанию (по умолчанию);
- `Desc` — по убыванию.

- ❑ Функция `ListLen(list [, delimiters])` возвращает длину списка.
- ❑ Функция `ListFirst(list [, delimiters])` возвращает первый элемент списка.
- ❑ Функция `ListLast(list [, delimiters])` возвращает последний элемент списка.
- ❑ Функция `ListRest(list [, delimiters])` возвращает список без первого значения.
- ❑ Функция `ListGetAt(list, position [, delimiters])` возвращает элемент из заданной позиции в списке.
- ❑ Функция `ListValueCount (list, value [, delimiters])` возвращает количество вхождений заданного значения в списке.
- ❑ Функция `ListValueCountNoCase (list, value [, delimiters])` возвращает количество вхождений заданного значения в списке без учета регистра.

Например:

```
<HTML>
<HEAD>
  <TITLE>myList</TITLE>
</HEAD>
<BODY>
  <CFSET myList =
    "Black,Magenta,Cyan,Orange,Darkgray,Pink,Gray,White,Lightgray,Yellow">
  <CFOUTPUT>List: #myList#</CFOUTPUT><br>
  <CFOUTPUT>Count of elements: #ListLen(myList)#</CFOUTPUT><br>
  <CFOUTPUT>The first element: #ListFirst(myList)#</CFOUTPUT><br>
  <CFOUTPUT>The last element: #ListLast(myList)#</CFOUTPUT><br>
  <CFOUTPUT>The third element: #ListGetAt(myList,3)#</CFOUTPUT><br>
  <CFOUTPUT>The element — "Orange" has number of a position:
    #ListFind(myList, "Orange")#</CFOUTPUT><br>
</BODY>
</HTML>
```

В результате приведенного выше примера на экран будет выведено следующее:

```
List: Black,Magenta,Cyan,Orange,Darkgray,Pink,Gray,White,Lightgray,Yellow
Count of elements: 10
The first element: Black
```

The last element: Yellow

The third element: Cyan

The element – "Orange" has number of a position: 4

Изменение и преобразование списков

- ❑ Функция `ListAppend(list, value [, delimiters])` добавляет значение в конец списка.
- ❑ Функция `ListPrepend(list, value [, delimiters])` добавляет значение в начало списка.
- ❑ Функция `ListInsertAt(list, position, value [, delimiters])` добавляет значение в заданную позицию списка.
- ❑ Функция `ListSetAt(list, position, value [, delimiters])` изменяет значение в заданной позиции списка.
- ❑ Функция `ListDeleteAt(list, position [, delimiters])` удаляет значение из заданной позиции.
- ❑ Функция `ListChangeDelims(list, new_delimiter [, delimiters])` заменяет в списке все разделители.
- ❑ Функция `ListQualify(list, qualifier [, delimiters] [, elements])` выделяет элементы списка с помощью указанного символа или набора символов. Здесь в качестве параметра `elements` может быть включен символ или набор символов, которые необходимо выделить.
- ❑ Функция `ListToArray(list [, delimiter])` возвращает массив, преобразуя данные из списка с учетом указанного разделителя.
- ❑ Функция `ValueList(query.column [, delimiter])` возвращает список значений колонки предварительно выполненного запроса.
- ❑ Функция `QuotedValueList(query.column [, delimiter])` возвращает список значений колонки предварительно выполненного запроса, с выделением элементов списка с помощью кавычек.

Например:

```
<HTML>
<HEAD>
  <TITLE>myList</TITLE>
</HEAD>
<BODY>
  <CFSET myList =
"Black,Magenta,Cyan,Orange,Darkgray,Pink,Gray,White,Lightgray,Yellow">
  <CFOUTPUT>List: #myList#</CFOUTPUT><br>
  <CFSET myColorArray = ListToArray(ListAppend(myList, "NewColor"))>
```

```
<CFLOOP INDEX = "i" FROM = "1" TO = #ArrayLen(myColorArray)#>
    <CFOUTPUT>#i#. #myColorArray[i]#</CFOUTPUT><br>
</CFLOOP>
</BODY>
</HTML>
```

В результате приведенного выше примера на экран будет выведено следующее:

List: Black,Magenta,Cyan,Orange,Darkgray,Pink,Gray,White,Lightgray,Yellow

1. Black
2. Magenta
3. Cyan
4. Orange
5. Darkgray
6. Pink
7. Gray
8. White
9. Lightgray
10. Yellow
11. NewColor

В предыдущих примерах, демонстрирующих обработку списков, мы определяли строковую переменную `myList` вручную. Впрочем, список также можно сформировать из предварительно выполненного запроса. К примеру, получим список круизов из уже знакомой нам базы данных `Cruise` и отсортируем полученные значения по алфавиту, используя функцию `ListSort`:

```
<HTML>
<HEAD>
    <TITLE>myList</TITLE>
</HEAD>
<BODY>
    <CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
        SELECT *
        FROM Tour
    </CFQUERY>
    <CFSET myListTour = ValueList(GetTour.Name_tou)>
    <CFOUTPUT>#ListSort(myListTour, "Text", "Asc")#</CFOUTPUT>
</BODY>
</HTML>
```


В результате работы приведенного примера на экран будет выведен список маршрутов через запятую, как показано ниже:

```
1-hr Seaport Liberty,2-hr Harbor Lights Evening,2-hr Semi Circle,The Full  
Island
```

Резюме

В данной главе вы познакомились с возможностями обработки массивов, структур и списков. Эти типы данных облегчают понимание ColdFusion и в дальнейшем позволят вам с легкостью разрабатывать собственные Web-приложения.



Разработка пользовательских тегов

На сегодняшний день в ColdFusion входит более семидесяти встроенных тегов, позволяющих создавать полноценные Web-приложения. Также сообщество разработчиков, использующих ColdFusion, предлагает набор пользовательских тегов для свободного распространения, с которыми можно ознакомиться и при необходимости скачать из Internet по адресу <http://devex.allaire.com/Developer/Gallery/CustomTags.cfm>.

Впрочем, почему бы нам не научиться создавать собственные пользовательские теги, тем более что это не требует громадных усилий. Пользовательский тег представляет собой некий код с возможностью его многократного использования, и поведение которого может зависеть от передаваемых атрибутов.

Применение пользовательских тегов позволяет ускорить процесс разработки Web-приложений и облегчить сопровождение будущего продукта.

Пользовательские теги в ColdFusion могут быть двух видов:

- ☐ теги, имена которых содержат приставку `CF_`, — это теги, разработанные непосредственно с использованием языка CFML и представляющие из себя файлы с расширением `cfm`;
- ☐ теги, имена которых начинаются с приставки `CFX_`, разработанные на Java или C++.

Создание CF-тегов

Как уже было отмечено, CF-теги создаются на основе CFML-страниц и их имена записываются с приставкой `CF_`. Например, для вызова тега, размещенного на странице `MyFirstTag.cfm`, необходимо использовать следующую конструкцию:

```
<CF_MyFirstTag>
```

Пользовательский тег может также записываться с указанием закрывающего тега, например:

```
<CF_MyFirstTag>
```

```
<TABLE border="0"><tr>
```

```
<TD bgcolor="Green"><FONT color="White">Hello!</FONT></TD>
</TR></TABLE>
</CF_MyFirstTag>
```

Размещать пользовательский тег следует либо там, где располагается страница, вызывающая его, либо в специально предусмотренном для этой цели каталоге. По умолчанию каталог для хранения пользовательских тегов находится на X:\Cfusion\CustomTags, где X — устройство, на которое был предустановлен ColdFusion Server. Но это не мешает изменять существующий путь или добавлять новые пути для размещения и в дальнейшем обнаружения системой пользовательских тегов. На рис. 14.1 показана страница администратора "ColdFusion Administrator" открытая в разделе **Server | Extensions | Custom Tag Paths** (Сервер | Расширения | Путь к пользовательским тегам), где предоставляется возможность регистрировать новые пути для пользовательских тегов.

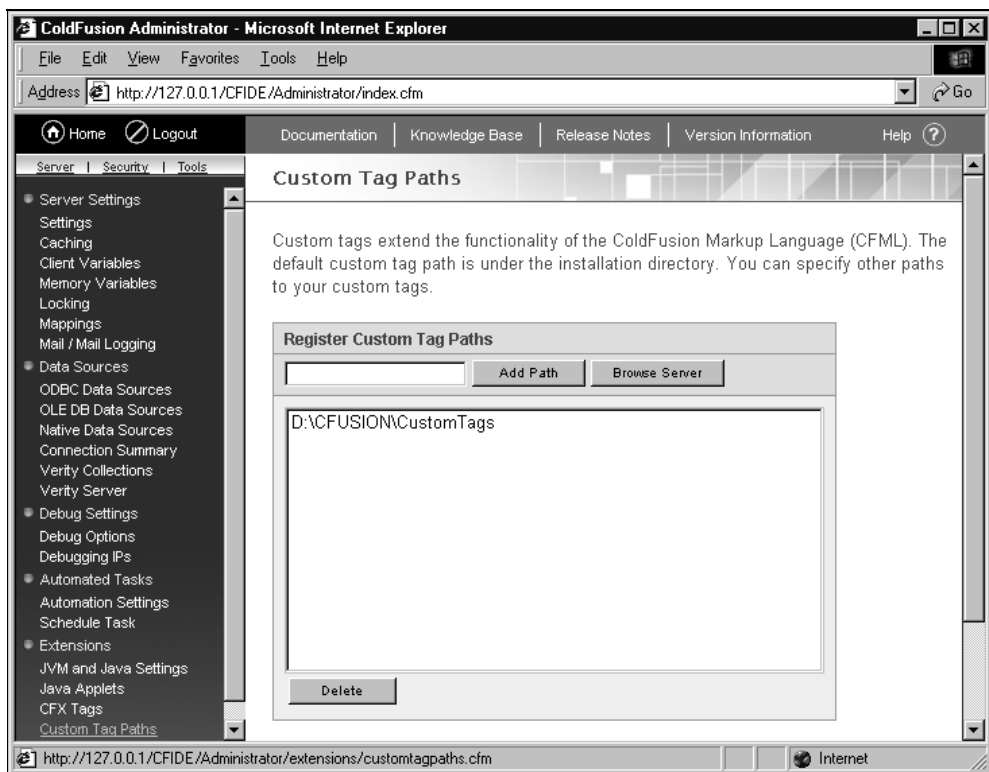


Рис. 14.1. Страница "ColdFusion Administrator", раздел **Custom Tag Paths**

При вызове тега также можно передавать значения переменных в качестве атрибутов. Например, предположим, что необходимо создать пользователь-

ский тег по преобразованию температуры воздуха из значений по шкале Цельсия (C) в градусы по Фаренгейту (F) и наоборот. Предварительно для данного примера определим синтаксис нового тега:

```
<CF_TemperatureTransform  
  Value="Number"  
  [CtoF="Boolean"]>
```

где Value — числовое значение, требуемое для преобразования; CtoF — логическое значение, указывающее направление преобразования. Причем атрибут Value является обязательным в отличие от атрибута CtoF. По умолчанию направление преобразования происходит из (C) в (F) впрочем, так же, как и при указании атрибута равным 1, Yes, True или просто при упоминании данного атрибута без указания значения.

Теперь, когда задача определена и сформулированы все условия, можно приступить к написанию страницы TemperatureTransform.cfm (листинг 14.1).

Листинг 14.1. Код страницы TemperatureTransform.cfm

```
<!--- Изменение температуры (Transformation of temperature) --->  
<CFIF isDefined("Attributes.Value")>  
  <CFIF isDefined("Attributes.CtoF")>  
    <CFIF isNumeric(Attributes.Value) AND isBoolean(Attributes.CtoF)>  
      <CFIF YesNoFormat(Attributes.CtoF) equal "Yes">  
        <CFSET Caller.Temperature = Attributes.Value*1.8+32 & " F">  
      <CFELSE>  
        <CFSET Caller.Temperature = (Attributes.Value-32)*5/9 & " C">  
      </CFIF>  
    <CFELSE>  
      <CFSET Caller.Temperature = "Error1">  
    </CFIF>  
  <CFELSE>  
    <CFSET Caller.Temperature = Attributes.Value*1.8+32 & " F">  
  </CFIF>  
<CFELSE>  
  <CFSET Caller.Temperature = "Error2">  
</CFIF>
```

После чего остается только использовать новый тег. Например, так:

```
<CF_TemperatureTransform Value = "21">  
<CFOUTPUT>Temperature: #Variables.Temperature#</CFOUTPUT>
```

В результате будет выведена следующая строка:

```
Temperature: 69.8 F
```

Если есть необходимость преобразования в обратную сторону, можно применить наш тег следующим образом:

```
<CF_TemperatureTransform Value = "69.8" CtoF="No">
<CFOUTPUT>Temperature: #Variables.Temperature#</CFOUTPUT>
```

Что позволит получить следующий результат:

```
Temperature: 21 C
```

В приведенном выше примере были использованы переменные атрибутов пользовательских тегов `Attributes`, позволяющие анализировать передаваемые атрибуты в теле пользовательского тега, и также применялись переменные источников вызова `Caller` для передачи переменных из шаблонов пользовательских тегов `ColdFusion`.

Также можно указывать вложение тегов, т. е. когда в вызываемом шаблоне пользовательского тега происходит обращение к другому пользовательскому тегу. Например, следующим шагом мы несколько модернизируем наш пример преобразования температуры.

В нашем примере при операции перевода значения температуры может возникнуть ситуация, когда значение после запятой достигнет немыслимых размеров. Предположим, если перевести по нашей формуле 70° по Фаренгейту в значение по шкале Цельсия, то в результате получим:

```
Temperature: 21.1111111111 C
```

Что является не очень удачным результатом. Для решения этой проблемы, в общем-то, достаточно воспользоваться существующей в `ColdFusion` функцией `NumberFormat()`. Однако мы "усложним" себе задачу и создадим еще один пользовательский тег для форматирования числовых величин (листинг 14.2).

Синтаксис пользовательского тега, позволяющего выполнить форматирование числовых величин, таков:

```
<CF_NumFormat
    Value = "Number"
    Format = "Mask">
```

где `Value` — числовое значение, требуемое для форматирования; `Format` — маска, определяющая формат и соответствующая правилу задания маски для функции `ColdFusion NumberFormat()`.

Листинг 14.2. Код страницы `NumFormat.cfm`

```
<CFIF isDefined("Attributes.Value") AND isDefined("Attributes.Format")>
    <CFSET Caller.Num = NumberFormat(Attributes.Value, Attributes.Format)>
```

```
<CFELSE>
    <CFSET Caller.Num = NumberFormat(0, "____.")>
</CFIF>
```

Теперь, когда мы обладаем таким замечательным пользовательским тегом, как `CF_NumFormat`, его можно использовать следующим образом:

```
<CF_NumFormat Value = "10.123" Format = "____">
<CFOUTPUT>#Variables.Num#</CFOUTPUT>
```

В результате получим число:

10.1

Далее представим обновленный код пользовательского тега для преобразования температуры (листинг 14.3), где используется вложение пользовательского тега `<CF_NumFormat>`.

Листинг 14.3. Код страницы `TemperatureTransform2.cfm`

```
<!--- Transformation of temperature Version2 --->
<CFSET Caller.Temperature = "">
<CFIF isDefined("Attributes.Value")>
    <CFIF isDefined("Attributes.CtoF")>
        <CFIF isNumeric(Attributes.Value) AND isBoolean(Attributes.CtoF)>
            <CFIF YesNoFormat(Attributes.CtoF) equal "Yes">
                <CFSET Val = Attributes.Value*1.8+32>
                <CFSET Measurement = " F">
            <CFELSE>
                <CFSET Val = (Attributes.Value-32)*5/9>
                <CFSET Measurement = " C">
            </CFIF>
        <CFELSE>
            <CFSET Caller.Temperature = "Error1">
        </CFIF>
    <CFELSE>
        <CFSET Val = Attributes.Value*1.8+32>
        <CFSET Measurement = " F">
    </CFIF>
<CFELSE>
    <CFSET Caller.Temperature = "Error2">
</CFIF>
```

```
<CFIF Caller.Temperature contains "Error">
<CFELSE>
    <CF_NumFormat Value = "#Val#" Format = "____.____" >
    <CFSET Caller.Temperature = #Variables.Num# & "#Measurement#">
</CFIF>
```

Возвращаясь к нашей проблеме, переведем все те же 70° по Фаренгейту в значение по шкале Цельсия, как показано ниже:

```
<CF_TemperatureTransform2 Value = "70" CtoF = "No">
<CFOUTPUT>Temperature: #Variables.Temperature#</CFOUTPUT>
```

В результате получим такой результат:

```
Temperature: 21.1 C
```

При использовании вложенных тегов применяются функции, позволяющие получать доступ к наследственным данным:

- ❑ `GetBaseTagList()` — возвращает список порожденных тегов через запятую в верхнем регистре;
- ❑ `GetBaseTagData(TagName, [InstanceNumber])` — возвращает объект, содержащий данные *n*-го предка. По умолчанию самый близкий предок.

К примеру, если тег `<CF_NumFormat>` дополнить строкой кода:

```
<CFOUTPUT>#GetBaseTagList()#</CFOUTPUT>
```

то при прямом обращении результатом будет являться значение, представленное ниже:

```
CFOUTPUT, CFIF, CF_NUMFORMAT
```

При обращении к тегу `<CF_NumFormat>` внутри пользовательского тега `<CF_TemperatureTransform2>` как к вложенному тегу, результатом будет значение:

```
CFOUTPUT, CFIF, CF_NUMFORMAT, CFELSE, CF_TEMPERATURETRANSFORM2
```

Обработав полученное значение, можно "вычлени" всех имеющихся "родителей" данного тега.

Как уже было отмечено в одном из первых примеров, пользовательский тег может записываться с указанием закрывающего тега, при этом ColdFusion позволяет отслеживать начало и окончание выполнения пользовательского тега. Определить же способ выполнения тега можно посредством переменной `ExecutionMode`, используя при этом относительную ссылку `ThisTag`, как показано в следующем примере:

```
<CFSWITCH EXPRESSION=#ThisTag.ExecutionMode#>
    <CFCASE VALUE = 'Start'>
        <!-- Начало тега --->
```

```
</CFCASE>
<CFCASE VALUE = 'End'>
    <!-- Конец тега --->
</CFCASE>
</CFSWITCH>
```

Здесь блок, условно назовем его *Start*, выполняется в начале тега, соответственно блок *End* выполняется при указании окончания тега.

Для демонстрации применения пользовательских тегов с указанием окончания приведем пример тега, позволяющего преобразовывать текст в табличную форму с возможностью изменения характеристик данной таблицы и осуществлять исправление некоторых грамматических ошибок, допущенных в тексте. Назовем этот тег `<CF_MyTable>`.

Синтаксис тега `<CF_MyTable>`:

```
<CF_MyTable
```

```
    Width = "Number"
    Border = "Number"
    CellPadding = "Number"
    CellSpacing = "Number"
    TBGColor = "Color"
    TColor = "Color"
    TSize = "Number"
    TText = "String">
```

Текст...

```
</CF_MyTable>
```

где:

- ❑ `Width` — числовое значение, требуемое для определения ширины таблицы;
- ❑ `Border` — числовое значение, определяющее толщину обрамления таблицы;
- ❑ `CellPadding` — числовое значение, определяющее промежуток между ячейками таблицы;
- ❑ `CellSpacing` — числовое значение, определяющее промежуток между границам ячеек и их содержимым;
- ❑ `TBGColor` — цвет фона заголовка;
- ❑ `TColor` — цвет текста заголовка;
- ❑ `TSize` — числовое значение, определяющее размер шрифта заголовка;
- ❑ `TText` — символьное значение, определяющее текст заголовка.

Далее представим код пользовательского тега `<CF_MyTable>` (листинг 14.4).

Листинг 14.4. Код страницы MyTable.cfm

```

<!-- My Table --->
<CFSWITCH EXPRESSION=#ThisTag.ExecutionMode#>
  <CFCASE VALUE = 'Start'>
    <!-- Начало тега (Start tag processing) --->
    <CFOUTPUT>
      <TABLE width="#Attributes.Width#" border="#Attributes.Border#"
        cellpadding="#Attributes.CellPadding#"
        cellspacing="#Attributes.CellSpacing#"
        bgcolor="#Attributes.TBGcolor#">
        <TR><TD align="center">
          <FONT color="#Attributes.TColor#" size="#Attributes.TSize#">
            #Attributes.TText#
          </FONT>
        </TD></TR>
      </TABLE>
    </CFOUTPUT>
  </CFCASE>
  <CFCASE VALUE = 'End'>
    <!-- Начало тега --->
    <CFOUTPUT>
      <TABLE width = "#Attributes.Width#"
        border = "#Attributes.Border#"
        cellpadding = "#Attributes.CellPadding#"
        cellspacing = "#Attributes.CellSpacing#">
        <TR><TD bgcolor = "#Attributes.TBGcolor#"></TD></TR>
        <TR><TD align = "right">
          Time: #TimeFormat(Now(), "HH:mm:ss")#
        </TD></TR>
      </TABLE>

      <CFSET ThisTag.GeneratedContent =
        "<table width = " & #Attributes.Width# &
        " border = " & #Attributes.Border# &
        " cellpadding = " & #Attributes.CellPadding# &
        " cellspacing = " & #Attributes.CellSpacing# &
        "><tr><td>" &

```

```
#REReplaceNoCase (thisTag.GeneratedContent, "(w|tt)he", "the", "ALL")# &
    "</td></tr></table>">
</CFOUTPUT>
</CFCASE>
</CFSWITCH>
```

В отличие от тега `<CF_TemperatureTransform2>`, рассмотренного ранее, мы не анализируем наличие необходимых атрибутов и удовлетворение определенных правил при указании значений этих атрибутов для пользовательского тега `<CF_MyTable>`, т. к. целью данного примера является демонстрация определения способа выполнения тега, и при желании данный тег всегда можно совершенствовать и совершенствовать. Обратите лишь ваше внимание на использование переменной `GeneratedContent`. В сочетании с относительной ссылкой `ThisTag` данная переменная позволяет определять содержание между началом и окончанием тега. Функция же `ColdFusion REReplaceNoCase()`, также используемая в приведенном теге, в данном контексте позволяет заменять словосочетания `'whe'` и `'tthe'` на `'the'` по всему тексту.

В качестве примера применения пользовательского тега `<CF_MyTable>` создадим файл следующего содержания:

```
<HTML>
<HEAD>
    <TITLE>Custom Tag</TITLE>
</HEAD>
<BODY>
    <CF_MyTable Width = "380" Border = "0" TBGColor = "Green"
        CellPadding = "1" CellSpacing = "1"
        TColor = "White" TSize = "4"
        TText = "Grand Central Depot">
        Grand Central Depot the predecessor of vhe present terminal,
        was whe first station opened in 1871 by vhe "Commodore".
    </CF_MyTable>
</BODY>
</HTML>
```

Здесь преднамеренно по тексту были допущены грамматические ошибки, обнаруживаемые тегом `<CF_MyTable>`. На рис. 14.2 показан результат применения пользовательского тега `<CF_MyTable>`.

Впрочем, для передачи значений атрибутов пользовательского тега можно также использовать зарезервированный атрибут `AttributeCollection`, где в качестве значения необходимо указывать заранее созданную *структуру* (structure). При этом ключи структуры воспринимаются как атрибуты поль-

зовательского тега, а значения ключей структуры играют роль значений этих атрибутов. К примеру:

```
<CFSET AttrbMyTable = StructNew()>
<CFSET AttrbMyTable.Width = "380">
<CFSET AttrbMyTable.Border = "0">
<CFSET AttrbMyTable.TBGColor = "Green">
<CFSET AttrbMyTable.CellPadding = "1">
<CFSET AttrbMyTable.CellSpacing = "1">
<CFSET AttrbMyTable.TColor = "White">
<CFSET AttrbMyTable.TSize = "4">
<CFSET AttrbMyTable.TText = "Grand Central Depot">
<CF_MyTable AttributeCollection = #AttrbMyTable#>
    Grand Central Depot the predecessor of tthe present terminal,
    was whe first station opened in 1871 by tthe "Commodore".
</CF_MyTable>
```

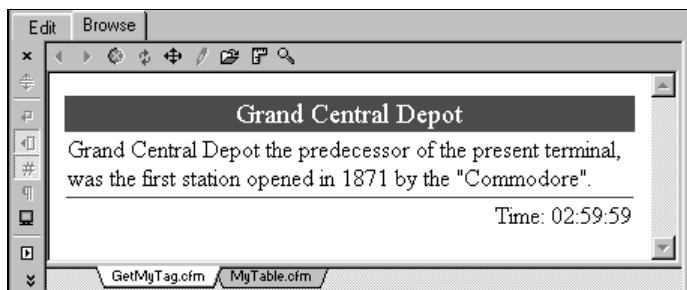


Рис. 14.2. Пример применения пользовательского тега `<CF_MyTable>`

Зарезервированный атрибут `AttributeCollection` можно применять в сочетании с пользовательскими атрибутами, например:

```
<CF_MyTable AttributeCollection = #AttrbMyTable# NewAttribute = "1">
```

Более подробно с понятием структур можно познакомиться в *главе 13*.

ColdFusion предоставляет еще один вариант для включения пользовательских тегов, уже с добавлением тега `<CFMODULE>`, где можно указывать полный путь размещения файла-шаблона.

Тег `<CFMODULE>`

Синтаксис:

```
<CFMODULE TEMPLATE = "template"
    NAME = "tag_name"
```

```
ATTRIBUTECOLLECTION = "collection_structure"
ATTRIBUTE_NAME1 = "value"
ATTRIBUTE_NAME2 = "value"
...>
```

В табл. 14.1 приведено описание атрибутов тега <CFMODULE>.

Таблица 14.1. Описание атрибутов тега <CFMODULE>

Атрибут	Описание
TEMPLATE	Шаблон. Используется вместо наименования. Определяет путь к CFML-файлу относительно текущей страницы
NAME	Наименование пользовательского тега. Применяется вместо шаблона в определенной форме. Например, если шаблон тега MyTable.cfm размещается в каталоге CustomTags\MyTags, тогда наименование пользовательского тега следует указать как MyTags.MyTable
ATTRIBUTECOLLECTION	Структура. Ключи структуры воспринимаются как атрибуты пользовательского тега, а значения ключей структуры играют роль значений передаваемых атрибутов. Необязательный атрибут
ATTRIBUTE_NAME	Атрибуты для пользовательского тега. Можно использовать необходимое количество атрибутов для определения нужных параметров для пользовательского тега. Необязательный атрибут

Проведем пример использования тега <CFMODULE> с применением указания шаблона:

```
<CFMODULE TEMPLATE = "MyTable.cfm"
  ATTRIBUTECOLLECTION = #AttribMyTable#>
  Grand Central Depot the predecessor of tthe present terminal,
  was whe first station opened in 1871 by tthe "Commodore".
</CFMODULE>
```

Создание CFX-тегов

В начале главы мы уже отметили, что CFX-теги разрабатываются на основе Java или C++. И если вас не удовлетворяет в полной мере язык CFML, то можно воспользоваться одним из упомянутых языков программирования для реализации пользовательских тегов, позволяющих расширить возможности собственного Web-приложения.

Давайте остановимся на Java, являющемся объектно-ориентированным языком программирования. Чтобы запустить программу на Java, необходимо ис-

пользовать интерпретатор Java, который будет выполнять инструкции откомпилированного байт-кода. Поскольку байт-код Java является архитектурно-нейтральным, программы на Java могут работать только на тех платформах, где реализована виртуальная машина Java (JVM, Java Virtual Machine), являющаяся интерпретатором и средством динамической поддержки.

На рис. 14.3 показана страница администратора "ColdFusion Administrator", открытая в разделе **Server | Extensions | JVM and Java Settings** (Сервер | Расширения | Параметры JVM и Java), где при использовании пользовательских тегов, разработанных на Java, необходимо указать месторасположение виртуальной машины в поле **Java Virtual Machine Path** (Путь к виртуальной машине Java).



Рис. 14.3. Страница "ColdFusion Administrator", раздел **JVM and Java Settings**

А также на странице администратора **JVM and Java Settings** (Параметры JVM и Java) устанавливаются:

- ☐ распределение области памяти, начальный — **Initial Heap Size** (Размер начальной области) и максимальный размер — **Max Heap Size** (Максимальный размер области);
- ☐ время загрузки виртуальной машины Java, где при указании переключателя **Load JVM when starting ColdFusion** (Загружать JVM при запуске

ColdFusion) JVM загружается при старте ColdFusion, в противном случае JVM будет загружена при первом запросе, который использует Java;

- ❑ список месторасположения классов **Class Path** (Путь к файлам классов), разделяемый точками с запятой (;) в среде Windows и двоеточиями (:) в среде UNIX.

Из числа устанавливаемых параметров на этой же странице, находящихся вне зоны видимости рис. 14.3, наиболее интересными для нас являются:

- ❑ месторасположение файла cfx.jar — **CFX Jar Path** (Путь к файлу cfx.jar), который содержит интерфейсы, используемые CFX-тегами, разработанные на основе языка Java. Значение по умолчанию: X:\CFusion\Java\classes, где X — устройство, на которое был предустановлен ColdFusion Server;
- ❑ месторасположение динамически загружаемых Java-классов — **Dynamic Class Load Path** (Путь к динамически загружаемым классам).

При формировании Java-кода, на основе которого предполагается создать CFX-тег, необходимо с использованием оператора `import` включать пакет `com.allaire.cfx`, расположенный в архиве `cfx.jar`. К примеру, создадим CFX-тег на основе Java, где объединяются значения трех входящих атрибутов A, B и C, а результат в виде строки передается на страницу, вызывающую пользовательский тег. Код данного примера приведен в листинге 14.5.

Листинг 14.5. Java-код MyTag.java

```
import com.allaire.cfx.* ;

public class MyTag implements CustomTag
{
    public void processRequest(Request request, Response response)
        throws Exception
    {
        String A = request.getAttribute("A");
        String B = request.getAttribute("B");
        String C = request.getAttribute("C");
        String ABC = toABC(A, B, C);
        response.write( "Result: " + ABC );
    }

    private String toABC(String a, String b, String c){
        String abc = a + b + c;
        return abc;
    }
}
```

Следующим шагом нам необходимо откомпилировать исходный текст Java из файла `MyTag.java` в байт-код Java-файла `MyTag.class`. Для подобных целей в Java существует программа `javac`. Автор предпочитает компилировать коды в байт-коды Java, используя обычную командную строку, как показано на рис. 14.4.

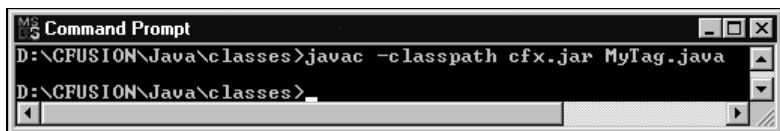


Рис. 14.4. Окно **Command Prompt** при выполнении компиляции `MyTag.java` в байт-код Java

Еще раз продемонстрируем команду для нашего примера:

```
javac -classpath cfx.jar MyTag.java
```

где параметр командной строки `-classpath` определяет путь, используемый `javac` для поиска классов, на которые имеются ссылки в исходном тексте. В нашем случае еще раз повторимся, это архив `cfx.jar`.

При отсутствии ошибок в исходном тексте в результате компиляции будет создан файл `MyTag.class`, который следует поместить по одному из указанных путей в поле **Class Path** (Путь к файлам классов) или в поле **Dynamic Class Load Path** (Путь к динамически загружаемым классам) раздела **Server | Extensions | JVM and Java Settings** (Сервер | Расширения | Параметры JVM и Java), размещенном на странице администратора. Например, на `D:\CFusion\CFX` или `D:\CFusion\CFX\Dynamic`. Размещая Java-классы в каталоге по одному из путей, указанных в поле **Class Path** (Путь к файлам классов), мы обрекаем их на статичное поведение. То есть при модификации откомпилированного Java-класса результат его выполнения будет неизменен до тех пор, пока ColdFusion Application Server не будет перезагружен. Впрочем, если существует необходимость использовать Java-класс как динамический, то его следует размещать по адресу, указанному в поле **Dynamic Class Load Path** (Путь к динамически загружаемым классам), что позволяет не перезагружать службу ColdFusion для отображения очередных изменений, произведенных по отношению к выполняемому Java-классу.

Заключительным действием для удачной работы тега CFX следует зарегистрировать созданный тег, используя все ту же страницу администратора ColdFusion, либо воспользовавшись системным реестром.

Для регистрации на странице администратора "ColdFusion Administrator" следует перейти в раздел **Server | Extensions | CFX Tags** (Сервер | Расширения | Теги CFX), нажать кнопку **Register Java CFX** (Регистрация

Java CFX) или кнопку **Register C++ CFX** (Регистрация C++ CFX) в зависимости от используемого языка программирования и заполнить предоставленную форму. На рис. 14.5 показана форма регистрации Java CFX с данными для приведенного примера.



Рис. 14.5. Форма регистрации Java CFX

Здесь:

- ❑ **Tag Name** (Имя тега) — наименование тега;
- ❑ **Class Name** (Имя класса) — наименование Java-класса без указания расширения class;
- ❑ **Description** (Описание) — описание тега.

После заполнения формы для подтверждения следует нажать одну из кнопок **Submit Changes** (Отправить изменения) — для удобства работы на странице расположены две кнопки **Submit Changes** (Отправить изменения) в верхней и в нижней части окна. На рис. 14.6 показан результат регистрации.



Рис. 14.6. Страница "ColdFusion Administrator", раздел **CFX Tags** после регистрации

Также для регистрации CFX-тегов можно использовать системный реестр. Например, для регистрации тега `<CFX_MyTag>` необходимо создать сценарий в следующей форме:

```
REGEDIT4
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Allaire\ColdFusion\CurrentVersion\CustomTags\CFX_MyTag]
```

```
"Description"=""
```

```
"Type"="java"
```

```
"JavaClass"="MyTag"
```

Затем приведенный сценарий следует сохранить в файле с расширением `reg`, что в последующем позволит зарегистрировать указанный CFX-тег посредством выполнения этого файла или с помощью команды импорта (**Import Registry File** (Импорт файла реестра)), запустив программу `Regedit` (Редактор реестра). Впрочем, с таким же успехом зарегистрировать CFX-тег можно вручную, открыв окно редактора реестра, как показано на рис. 14.7.

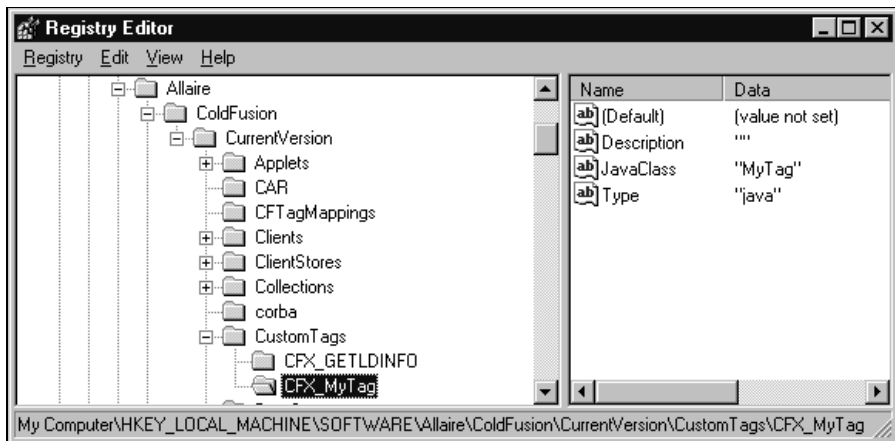


Рис. 14.7. Окно **Registry Editor** с зарегистрированным CFX-тегом

И только теперь, когда CFX-тег успешно создан и зарегистрирован, можно приступить к демонстрации примера вызова этого тега. Далее представим пример, позволяющий использовать разработанный нами тег `<CFX_MyTag>` на основе `MyTag.class`:

```
<HTML>
<HEAD>
  <TITLE>CFX_MyTag</TITLE>
</HEAD>
<BODY>
  <CFX_MyTag A = "Aa" B = "Bb" C = "Cc">
</BODY>
</HTML>
```

Результатом выполнения приведенного примера будет вывод следующей строки:

```
Result: AaBbCc
```

Вернемся к нашему примеру `MyTag.java` (см. листинг 14.5). В нем ключевыми действиями является использование встроенных объектов `Request` и `Response`. Объект `Request` служит для получения значений запроса, посланного со стороны страницы. К примеру, следующий код определяет строковую переменную `A` и присваивает ей значение передаваемого атрибута с аналогичным именем, используя метод `getAttribute`:

```
String A = request.getAttribute("A");
```

Объект `Response` имеет обратное назначение и предназначен для передачи информации вызываемой странице. Например, представленный ниже код отображает на странице клиента значение, определяемое с помощью метода `write`.

```
response.write("Result: " + ABC);
```

В свою очередь встроенный объект `Request` имеет следующие методы:

- ❑ `attributeExists` — проверяет существование атрибута;
- ❑ `debug` — проверяет существование атрибута отладки `Debug`;
- ❑ `getAttribute` — получает значение, передаваемое атрибутом;
- ❑ `getAttributeList` — получает список значений всех атрибутов;
- ❑ `getIntAttribute` — получает значение, передаваемое атрибутом как целое число;
- ❑ `getQuery` — получает передаваемый запрос;
- ❑ `getSetting` — получает значение установки пользовательского тега.

Встроенный объект `Response` также имеет набор методов:

- ❑ `addQuery` — добавляет запрос к вызываемой странице;
- ❑ `setVariable` — устанавливает переменную к вызываемой странице;
- ❑ `write` — отображает текст на вызываемой странице;
- ❑ `writeDebug` — отображает текст потока отладки (`debug stream`) на вызываемой странице.

Стоит отметить, что помимо объектов `Request` и `Response` существует еще один встроенный объект — `Query`, обеспечивающий интерфейс для работы с запросами ColdFusion. Мы же ограничимся только перечислением методов для этого объекта:

- ❑ `addRows` — добавляет новую строку к запросу;
- ❑ `getColumn` — получает количество колонок запроса;
- ❑ `getData` — получает элемент данных из запроса;
- ❑ `getName` — получает имя запроса;
- ❑ `getRowCount` — получает количество записей в запросе;
- ❑ `setData` — устанавливает элемент данных в пределах запроса.

Мы не рассматриваем в этой главе примеры создания CFX-тегов с помощью языка C++. Только отметим, что при использовании программного продукта Microsoft Visual C++ в среде Windows вы можете запускать так называемые мастера для создания CFX-тегов (ColdFusion Tag Wizard). При этом должен быть установлен комплект CFXAPI Tag Development Kit. Мастер генерирует DLL-файлы с основной структурой тега, содержащего единственную процедуру. Изменяя и тестируя этот тег, вы сможете быстро научиться работать в среде API.

Резюме

Подведем итог. В данной главе вы познакомились с CFML-тегом `<CFMODULE>` и CFML-функцией `REReplaceNoCase`.



Обработка ошибок

ColdFusion обладает достаточным набором инструментов и методов, позволяющих отлаживать и обрабатывать ситуации, которые приводят к ошибкам (исключительным ситуациям), и благодаря этим инструментам и методам можно предотвращать возможные ошибки и восстанавливать нормальную работу приложения после сбоев, которые в противном случае были бы фатальными.

Следует отметить, что главным инструментом при отладке CFML-страниц является непосредственно сам редактор ColdFusion Studio, работа с которым была описана в предыдущих главах книги. Впрочем, ColdFusion Server обладает некоторым набором инструментов, позволяющих отслеживать значения переменных, анализировать результаты выполнения запросов и отлаживать отдельные страницы.

Отладка приложения

Для отслеживания значения переменных можно использовать тег `<CFDUMP>`, позволяющий наглядно демонстрировать значения простых переменных, запросов, массивов, структур и WDDX-переменных, созданных с помощью тега `<CFWDDX>`.

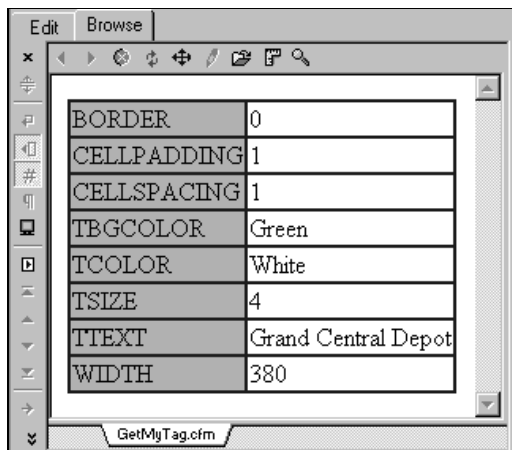
Например, для отладки структуры `AttribMyTable` достаточно добавить следующую строку:

```
<CFDUMP var = #AttribMyTable#>
```

В результате будет отображена таблица со значениями данной структуры, как показано на рис. 15.1.

Для отладки запроса (query) можно также использовать тег `<CFDUMP>`, позволяющий выводить таблицу значений указанного запроса. Однако существует еще одна разновидность отладки запроса. Это добавление атрибута `DEBUG`. Например:

```
<CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#" DEBUG="Yes">  
    SELECT * FROM Tour  
</CFQUERY>
```



BORDER	0
CELLPADDING	1
CELLSPACING	1
TBGCOLOR	Green
TCOLOR	White
TSIZE	4
TTEXT	Grand Central Depot
WIDTH	380

Рис. 15.1. Результат отладки структуры `AttribMyTable`

В результате внизу страницы будет выведена информация о количестве записей текущего запроса, времени выполнения и самом запросе, как показано ниже:

GetTour (Records=4, Time=10ms)

SQL = SELECT * FROM Tour

Для отладки отдельного документа с возможностью просмотра сведений о времени выполнения (execution time), параметрах URL (URL parameters) и переменных окружения CGI (CGI variables) можно воспользоваться простым добавлением в конец строки URL определения следующего параметра `mode=debug`. При этом вся строка URL для документа `MyPage.cfm` может выглядеть так:

`http://127.0.0.1/my-app/MyPage.cfm?mode=debug`

Немаловажными для отладки документа подобным образом, так же, как и для предыдущего примера с добавлением к запросу атрибута `DEBUG`, являются настройки на странице администратора "ColdFusion Administrator". По умолчанию возможность просмотра отладочной информации предоставляется всем. Для ограничения же необходимо определить IP-адреса компьютеров, имеющих право для просмотра отладочной информации, используя при этом раздел страницы администратора **Server | Debug Settings | Debugging IPs** (Сервер | Параметры отладки | IP-адреса для отлаки), как показано на рис. 15.2.

Также можно настроить получаемую информацию, воспользовавшись разделом **Server | Debug Settings | Debug Options** (Сервер | Параметры отладки | Настройка отладки) все той же страницы администратора.



Рис. 15.2. Страница "ColdFusion Administrator", раздел **Debugging IPs**

Обработка ошибок

Методы, позволяющие обрабатывать ошибки, можно разделить на две категории:

- ❑ обработка ошибок через конфигурационный файл приложения `Application.cfm` с применением тега `<CFERROR>`, позволяющего перенаправлять сообщения об ошибках на собственные шаблоны;
- ❑ формирование конструкций обработки исключительных ситуаций непосредственно на страницах приложения с использованием тегов `<CFTRY>/<CFCATCH>`, `<CFTHROW>`, `<CFRETHROW>`.

Тег `<CFERROR>`

Синтаксис:

```
<CFERROR TYPE = "Validation | Request | Monitor | Exception"
          TEMPLATE = "template_path"
          MAILTO = "email_address"
          EXCEPTION = "exception_type">
```

В табл. 15.1 приведено описание атрибутов тега `<CFERROR>`.

Таблица 15.1. Описание атрибутов тега `<CFERROR>`

Атрибут	Описание
TYPE	Тип ошибки. Обязательный атрибут. Возможные значения: <code>Validation</code> — происходит при неправильном вводе данных в форме; <code>Request</code> — происходит при любых исключениях, которые иначе не обрабатываются. При этом шаблон сообщения об ошибках выполняется после исполнения CFML-страницы. Следует использовать в резервных целях; <code>Monitor</code> — происходит перед любыми другими обработками исключений, после чего обработка ошибок может продолжаться иными средствами, например с помощью тегов <code><CFTRY>/<CFCATCH></code>. Для данного типа можно определять любые возможные исключения. Используется для контроля и отладки исключений только на этапе проектирования приложения; <code>Exception</code> — используется для определения исключительной ситуации. Можно определить различные шаблоны ошибок для разных типов исключений
TEMPLATE	Относительный путь к пользовательскому шаблону ошибок. Обязательный атрибут
MAILTO	Электронный адрес, по которому следует уведомлять о происходящих ошибках. Необязательный атрибут
EXCEPTION	Исключение. Обязательный атрибут при использовании таких типов ошибок, как <code>Monitor</code> и <code>Exception</code>

Типы исключительных ситуаций

Тип исключительной ситуации может быть учтен как при использовании атрибута `EXCEPTION` тега `<CFERROR>`, так и в сочетании с атрибутом `TYPE` тега `<CFCATCH>`, предназначенным для формирования конструкций обработки исключительных ситуаций, непосредственно на страницах приложения. Основные исключительные ситуации ColdFusion представлены в табл. 15.2.

Таблица 15.2. Типы исключительных ситуаций

Тип	Описание
Database	Фиксирует неудачные действия с базами данных и проблемы, связанные с ODBC
Template	Обнаруживает ошибки в работе с шаблонами. Подобные исключения могут генерироваться при использовании тегов <code><CFINCLUDE></code> , <code><CFMODULE></code> и <code><CFERROR></code>

Таблица 15.2 (окончание)

Тип	Описание
MissingInclude	Обнаруживает ошибки, где встречаются ссылки на несуществующие файлы
Object	Обнаруживает ошибки в программном коде ColdFusion, который предусматривает работу с объектами
Security	Фиксирует ошибки в программном коде ColdFusion, который предусматривает обеспечение безопасности
Expression	Обнаруживает ошибки при неправильном использовании выражений
Lock	Фиксирует ошибки при неудачной блокировке или во время выполнения блокировок, организованных с помощью тега <CFLock>
Application	Обнаруживает ошибки приложения, вызываемые тегом <CFTHROW>
Any	Обнаруживает любые ошибки, которые не определены другими исключениями, в том числе неожиданные внутренние ошибки
Custom_exception_type	Обнаруживает пользовательские ошибки, вызываемые тегом <CFTHROW>

Создание собственных шаблонов сообщений об ошибках

Для обнаружения большинства ошибок ColdFusion Server использует стандартную страницу. Впрочем, определяя собственные шаблоны в конфигурационном файле приложения Application.cfm, вы можете управлять выводимой информацией при возникновении ошибок различного характера. Причем ColdFusion позволяет определять шаблоны для различных типов исключительных ситуаций и обращаться с разными типами исключений различными способами.

Предположим, нам необходимо выводить сообщения об ошибках двух видов для ситуаций при некорректном заполнении форм (Validation) и для любых других ошибочных ситуаций, которые иначе не обрабатываются (Request). Решением данной задачи может быть следующий код, помещенный в конфигурационный файл Application.cfm приложения:

```
<CFERROR TYPE="Validation"
    TEMPLATE="ErrorsTemplate1.cfm"
    MAILTO="Administrator@ipr.ru">
```



```
<CFERROR TYPE="Request"
      TEMPLATE="ErrorsTemplate2.cfm"
      MAILTO="WebMaster@ipr.ru">
```

Здесь ErrorsTemplate1.cfm и ErrorsTemplate2.cfm являются шаблонами, определяющими содержание сообщений при возникновении исключительных ситуаций. На рис. 15.3 показана условная схема вызова шаблонов сообщений об ошибках.

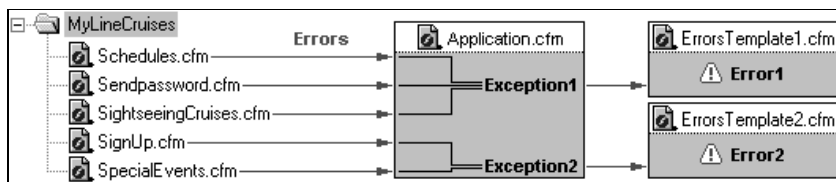


Рис. 15.3. Схема вызова шаблонов сообщения об ошибках

Для формирования непосредственно самих шаблонов сообщения об ошибках ColdFusion предоставляет целый набор встроенных переменных, определяющих ошибки. Причем в зависимости от типа ошибки могут быть активированы разные переменные. Например, для типа ошибок `Validation` поместим в наш шаблон ErrorsTemplates1.cfm следующие переменные:

- ❑ `Error.ValidationHeader` — текст заголовка;
- ❑ `Error.InvalidFields` — список возможных ошибок, включает значения полей или текст, определяемый атрибутом `VALUE` тега `<INPUT>` для скрытых (hidden) полей;
- ❑ `Error.ValidationFooter` — текст подзаголовка.

А для типа ошибок `Request` поместим в шаблон ErrorsTemplates2.cfm такие встроенные переменные:

- ❑ `Error.Template` — выполняемая страница в момент возникновения ошибки;
- ❑ `Error.DateTime` — дата и время, когда произошла ошибка;
- ❑ `Error.Diagnostics` — детальная диагностика ошибки от ColdFusion Server;
- ❑ `Error.Browser` — обозреватель, который запущен в момент возникновения ошибки;
- ❑ `Error.RemoteAddress` — IP-адрес удаленного клиентского компьютера;
- ❑ `Error.MailTo` — адрес электронной почты, по которому следует уведомлять о произошедшей ошибке. Соответствует значению атрибута `MAILTO` тега `<CFERROR>`.

Также можно использовать следующие встроенные переменные ошибок:

- ❑ `Error.Type` — тип возникшей ошибки;
- ❑ `Error.HTTPReferer` — страница, с которой клиент обратился к связанной странице, где произошла ошибка;
- ❑ `Error.QueryString` — передаваемые параметры в строке URL в момент возникновения ошибки.

Далее продемонстрируем полный код страниц `ErrorsTemplate1.cfm` (листинг 15.1) и `ErrorsTemplate2.cfm` (листинг 15.2), позволяющих выводить сообщения в момент возникновения различных ошибок.

Листинг 15.1. Код страницы `ErrorsTemplate1.cfm`

```
<HTML>
<HEAD>
  <link rel="STYLESHEET" type="text/css" href="Style.css">
  <TITLE>ErrorsTemplate1</TITLE>
</HEAD>
<BODY>
  <h1>Error1</h1>
  <TABLE border="0">
    <TR><TD bgcolor="Black" class="txtErr">ValidationHeader</TD></TR>
    <TR><TD bgcolor="Silver">#Error.ValidationHeader#</TD></TR>
    <TR><TD bgcolor="Black" class="txtErr">InvalidFields</TD></TR>
    <TR><TD bgcolor="Silver">#Error.InvalidFields#</TD></TR>
    <TR><TD bgcolor="Black" class="txtErr">ValidationFooter</TD></TR>
    <TR><TD bgcolor="Silver">#Error.ValidationFooter#</TD></TR>
  </TABLE>
</BODY>
</HTML>
```

Листинг 15.2. Код страницы `ErrorsTemplate2.cfm`

```
<HTML>
<HEAD>
  <link rel="STYLESHEET" type="text/css" href="Style.css">
  <TITLE>ErrorsTemplate2</TITLE>
</HEAD>
<BODY>
  <h1>Error2</h1>
```

```

<TABLE border="0">
  <TR bgcolor="Black">
    <TD class="txtErr">Template</td>
    <TD class="txtErr">DateTime</td>
  </TR>
  <TR bgcolor="Silver">
    <TD>#Error.Template#</td>
    <TD>#Error.DateTime#</td>
  </TR>
  <TR><TD colspan="2" bgcolor="Black"
    class="txtErr">Diagnostics</td></tr>
  <TR><TD colspan="2" bgcolor="Silver" class="txtErr2">
    #Error.Diagnostics#</TD></TR>
  <TR><TD colspan="2" bgcolor="Black"
    class="txtErr">Browser</TD></TR>
  <TR><TD colspan="2" bgcolor="Silver">#Error.Browser#</TD></TR>
  <TR bgcolor="Black">
    <TD class="txtErr">RemoteAddress</td>
    <TD class="txtErr">MailTo</td>
  </TR>
  <TR bgcolor="Silver">
    <TD>#Error.RemoteAddress#</TD>
    <TD>#Error.MailTo#</TD>
  </TR>
</TABLE>
</BODY>
</HTML>

```

В дальнейшем остается лишь проверить работу созданных нами шаблонов. Например, создадим форму ввода, где при заполнении полей определенно будет сгенерирована ошибка с типом Validation:

```

<HTML>
  <HEAD>
    <TITLE>ErrorValidation</TITLE>
  </HEAD>
  <BODY>
    <FORM action="Action.cfm" method="Post">
      <input type="Text" name="Field"><br>
      <input type="Text" name="Field_integer">

```

```
<input type="Hidden" name="Field_integer"
      value="Error hidden field">
<input type="Submit" value="Send">
</FORM>
</BODY>
</HTML>
```

На рис. 15.4 показана форма описанного выше кода с введенными значениями полей, приводящими к фатальному исходу.

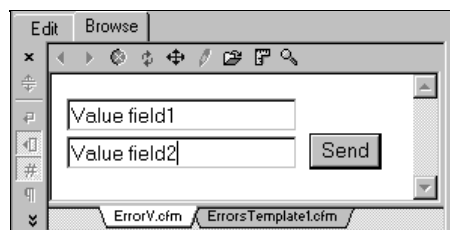


Рис. 15.4. Пример, приводящий к ошибке с типом Validation

После подтверждения введенных значений в представленной форме будет выведен шаблон, предупреждающий об ошибке (рис. 15.5).

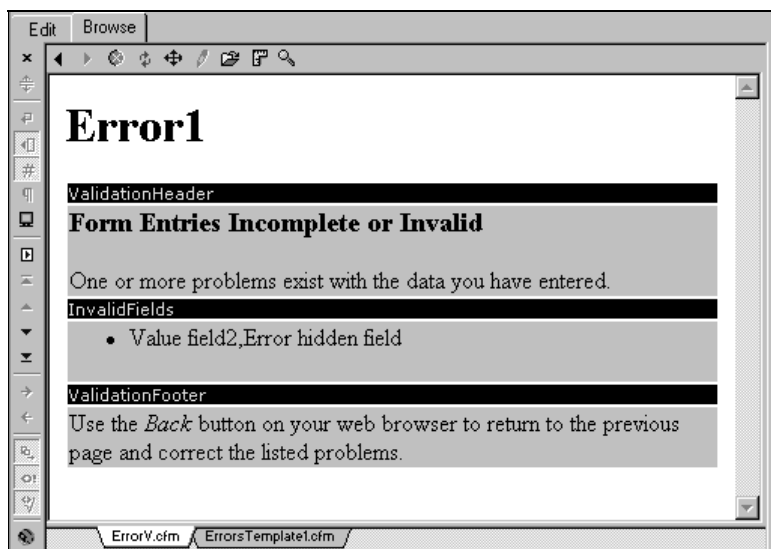


Рис. 15.5. Шаблон предупреждения об ошибке с типом Validation

Приведем еще один пример, где в коде преднамеренно допустим ошибку типа Request, как показано ниже:

```
<CFIF isDeeffined("MyVal")>
```

```
</CFIF>
```

Это приведет к выводу на экран шаблона предупреждения об ошибке с типом Request, как это показано на рис. 15.6.

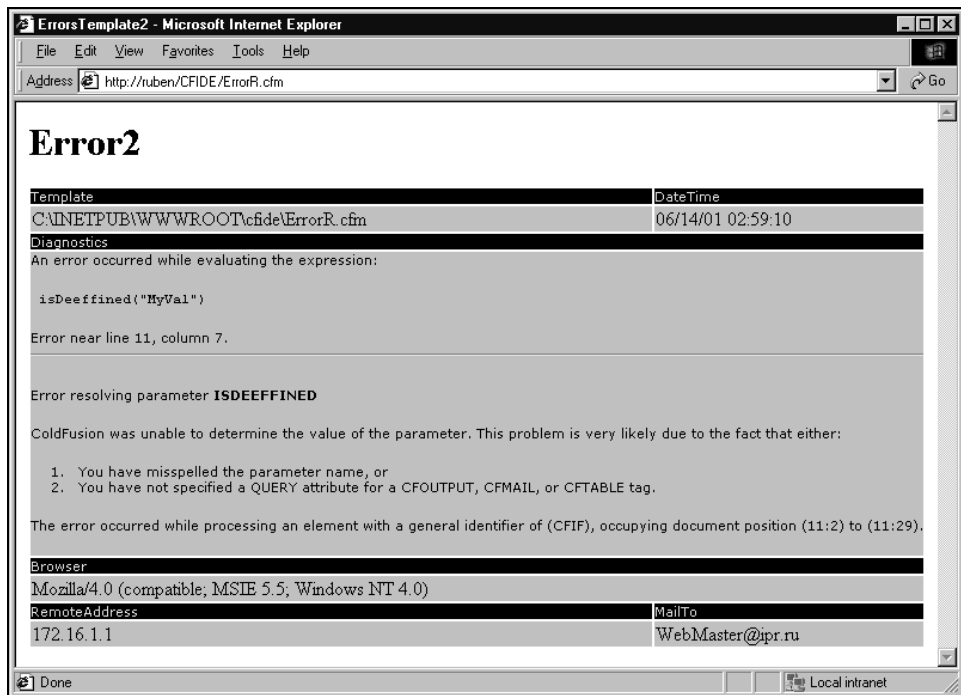


Рис. 15.6. Шаблон предупреждения об ошибке с типом Request

Пользовательские шаблоны сообщений об ошибках позволяют только обогатить Web-приложения собственными стилизованными формами. Необходимо отметить, что ColdFusion Server автоматически регистрирует ошибки в log-файлах в следующих случаях:

- ☐ при обработке ошибок по умолчанию;
- ☐ при использовании тега `<CFERROR>` с типом Request, позволяющего производить обработку ошибок при любых исключениях, которые иначе не обрабатываются.

В других случаях, при необходимости регистрировать ситуации, приводящие к ошибкам, необходимо создавать пользовательский log-файл и формировать код, позволяющий производить запись о возникающих ошибках.

Обработка исключительных ситуаций

Одно из главных преимуществ системы обработки исключительных ситуаций по сравнению с традиционными методами обработки ошибок состоит в отделении процедуры обнаружения ошибки от процедуры ее обработки. В ColdFusion для формирования конструкций обработки исключительных ситуаций непосредственно на страницах приложения необходимо использовать теги `<CFTRY>/<CFCATCH>`.

Теги `<CFTRY>/<CFCATCH>`

```
<CFTRY>
```

Код с возможным исключением

```
<CFCATCH TYPE = "ExceptionType">
```

Код обработки исключительной ситуации

```
</CFCATCH>
```

Дополнительный код для блока cfcatch

```
</CFTRY>
```

Здесь атрибут `TYPE` может принимать одно из значений типа исключительной ситуации, описанных выше. Внутри же блока `<CFTRY>` содержится код, в ходе выполнения которого могут возникать исключения или аварийное завершение программы, соответствующим образом обрабатываемые посредством блока `<CFCATCH>` или путем переадресации исключения с использованием тега `<CFTHROW>`.

Например:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>CFTRY example1</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<CFTRY>
```

```
<CF_NumFormat Value = "10.123" Format = "__._e">
```

```
<CFCATCH>
```

```
Error Message
```

```
</CFCATCH>
```

```
</CFTRY>
```

```
</BODY>
```

```
</HTML>
```

Здесь в коде программы между тегами `<CFTRY>` и `<CFCATCH>` преднамеренно допущена ошибка, а в блок `<CFCATCH>` помещен код обработки исключи-

тельной ситуации. В результате на экран будет выведено следующее сообщение, сигнализирующее об ошибке:

Error Message

Отметим, что в приведенном примере блок <CFCATCH> использован без указания типа исключительной ситуации, что подразумевает обработку любых ошибок. Причем внутри блока <CFTRY> может следовать любое количество блоков <CFCATCH>. При отсутствии же хотя бы одного блока <CFCATCH>, что вполне допустимо, применение блока <CFTRY> бессмысленно.

Следующим примером разделим обработку исключений для разных типов, в частности предусмотрим исключение, фиксирующее неудачные действия по отношению к базам данных (Database), при неправильном использовании выражений (Expression) и для остальных ошибок (Any):

```
<HTML>
<HEAD>
  <TITLE>CFTRY example2</TITLE>
</HEAD>
<BODY>
  <CFTRY>
    <CFQUERY NAME="GetTour" DATASOURCE="#application.dsn#">
      SELECT * FROM Tour
    </CFQUERY>
    <CF_NumFormat Value = "10.123" Format = "__._e">
    <CFCATCH TYPE="Database">
      <CFOUTPUT>Error Code: #Cfcatch.ErrorCode#</CFOUTPUT><br>
      <CFOUTPUT>Detail: #Cfcatch.Detail#</CFOUTPUT>
    </CFCATCH>
    <CFCATCH TYPE="Expression">
      Expression Error
    </CFCATCH>
    <CFCATCH TYPE="Any">
      Any Error
    </CFCATCH>
  </CFTRY>
</BODY>
</HTML>
```

Все исключения обрабатываются последовательно в зависимости от расстановки блоков <CFCATCH>. В нашем примере, если проблем с базой данных не будет, то программа перейдет к обработке ошибок, возникших при использовании неправильных выражений. Если и здесь проблем не будет, то

следующим шагом станут обрабатываться все остальные возможные исключения. Причем в нашем случае при возникновении исключения с типом Database на экран будет выведен код и детальная информация об ошибке, определяемые с помощью внутренних переменных захвата Cfcatch.ErrorCode и Cfcatch.Detail. Например, следующий результат может указывать на невозможность соединения с базой данных:

```
Error Code: 08001
Detail: [Sybase][ODBC Driver][Adaptive Server Anywhere]Unable to connect
to database server: Database server not running
```

Приложение ColdFusion позволяет объявлять собственные исключительные ситуации и производить переадресацию исключений с использованием тега <CFTHROW>.

Тег <CFTHROW>

Синтаксис:

```
<CFTHROW TYPE = "exception_type"
          MESSAGE = "message"
          DETAIL = "detail_description"
          ERRORCODE = "error_code"
          EXTENDEDINFO = "additional_information">
```

В табл. 15.3 представлено описание атрибутов тега <CFTHROW>.

Таблица 15.3. Описание атрибутов тега <CFTHROW>

Атрибут	Описание
TYPE	Пользовательский тип исключения. Необязательный атрибут. Если данный тип не определен, то с помощью тега <CFCATCH> необходимо определить захват типа Application или Any, куда будет перенаправлено данное исключение
MESSAGE	Сообщение, которое описывает исключительную ситуацию. Обязательный атрибут
DETAIL	Детальное описание исключения. Необязательный атрибут
ERRORCODE	Пользовательский код ошибки. Необязательный атрибут
EXTENDEDINFO	Дополнительное описание исключения. Необязательный атрибут

Например, следующий код позволяет генерировать пользовательское исключение MyExp по воскресеньям:

```
<HTML>
<HEAD>
  <TITLE>CFTRY example3</TITLE>
```



```

</HEAD>
<BODY>
  <CFTRY>
    <CFSET NowDayWeek = DayOfWeek(Now())>
    <CFIF Variables.NowDayWeek eq 1>
      <CFTHROW TYPE="MyExp" ERRORCODE="1" EXTENDEDINFO="NotSundays"
        DETAIL="Today Sunday, that obliges you well to have a rest!"
        MESSAGE = "I do not work on Sundays.">
    </CFIF>
    <CFCATCH TYPE="MyExp">
      Message: <CFOUTPUT>#Cfcatch.Message#</CFOUTPUT>
    </CFCATCH>
    <CFCATCH TYPE="Any">
      Any Error
    </CFCATCH>
  </CFTRY>
</BODY>
</HTML>

```

И если предположить, что данный код выполнится в первый день недели, то на экран будет выведено сообщение об ошибке с помощью внутренней переменной захвата `Cfcatch.Message`, как показано ниже:

Message: I do not work on Sundays.

Если вам необходимо выполнить некоторые действия при возникновении исключения во внутреннем блоке `<CFTRY>`, но не прекратить на этом обработку, а передать исключение дальше, внешнему обработчику, вы можете воспользоваться тегом `<CFRETHROW>`, позволяющим регенерировать исключение. Например:

```

<HTML>
  <HEAD>
    <TITLE>CFTRY example4</TITLE>
  </HEAD>
  <BODY>
    <CFTRY>
      <CFTRY>
        <CF_NumFormat Value = "10.123" Format = "__._e">
        <CFCATCH TYPE="Expression">
          ErrNumber: <CFOUTPUT>#Cfcatch.ErrNumber#</CFOUTPUT><br>
        <CFRETHROW>

```

```
</CFCATCH>
</CFTRY>
<CFOUTPUT>MyCode</CFOUTPUT>
<CFCATCH TYPE="Any">
    Type: <CFOUTPUT>#Cfcatch.Type#</CFOUTPUT>
</CFCATCH>
</CFTRY>
</BODY>
</HTML>
```

В результате выполнения приведенного примера будет выведен номер ошибки и тип исключения, определяемые с помощью внутренних переменных захвата `Cfcatch.ErrNumber` и `Cfcatch.Type`, как показано ниже:

```
ErrNumber: 16399
```

```
Type: EXPRESSION
```

Чтобы подвести итог в описании исключительных ситуаций, приведем список внутренних переменных захвата, часть из которых уже была использована в представленных примерах.

Внутренние переменные захвата для большинства типов исключений:

- ❑ `Cfcatch.Type` — тип исключения;
- ❑ `Cfcatch.Message` — сообщение, которое описывает исключительную ситуацию;
- ❑ `Cfcatch.Detail` — детальное описание исключения;
- ❑ `Cfcatch.ExtendedInfo` — дополнительное описание исключения;
- ❑ `Cfcatch.ErrorCode` — код возникшей ошибки;
- ❑ `Cfcatch.TagContext` — содержит наименование тега (ID), наименование страницы, включая полный путь (Template) и позицию (Line и Column) расположения каждого тега в стековой последовательности тегов вплоть до тега исключения. Эта переменная может быть использована при установке флажка **Enable CFML Stack Trace** (Использовать стековую последовательность CFML), расположенного на странице "ColdFusion Administrator" в разделе **Server | Debug Settings | Debug Options** (Сервер | Параметры отладки | Настройка отладки). Данную переменную следует обрабатывать как массив.

Переменные захвата для исключения Database:

- ❑ `Cfcatch.NativeErrorCode` — собственный код ошибки, при неопределенности принимает значение, равное -1;
- ❑ `Cfcatch.SQLState` — SQL-выражение, при отсутствии SQL-выражения принимает значение, равное -1.

Переменная захвата для исключения `Expression: Cfcatch.ErrNumber` — номер ошибки выражения.

Переменные захвата для исключения `Lock`:

❑ `Cfcatch.LockName` — имя блокировки;

❑ `Cfcatch.LockOperation` — действие, которое потерпело неудачу.

Переменная захвата для исключения `MissingInclude:`

`Cfcatch.MissingFileName` — имя отсутствующего файла.

Резюме

Подведем итог. В данной главе вы познакомились с CFML-тегами `<CFERROR>`, `<CFTRY>/<CFCATCH>`, `<CFTHROW>`, `<CFRETHROW>`.



Рассылка и получение почты

Служба электронной почты настолько привычна сегодня, что многие просто не представляют, какой была бы жизнь в отсутствие этого полезного и неприхотливого способа связи. По заявлениям в Internet первое электронное письмо в мире было отправлено в 1971 году при тестировании нового, только что разработанного способа передачи текстовой информации между двумя компьютерами. Что касается содержания первого электронного письма, то остается только догадываться, каким оно было. Впрочем, это не столь важно. Главное, что такое событие позволило открыть еще один способ общения между людьми.

Существует множество программ, позволяющих "прогрессивному" сообществу общаться между собой. Здесь могут быть использованы любые технологии — от систем обмена сообщениями типа ICQ до видеоконференций. Несмотря на это, сегодня, как и на заре развития Internet, самым распространенным и наиболее надежным способом сообщить что-либо другому человеку или переслать файл остается электронная почта (e-mail).

Для работы с почтой можно использовать специализированную программу — почтовый клиент, или же зарегистрироваться в системе Web-почты и оперировать с почтой через браузер. Если пользователь ведет активную переписку, то второй способ практически неприменим: требования к почтовой программе для активных пользователей весьма высоки. ColdFusion в свою очередь позволяет оперировать почтой с помощью тегов `<CFMAIL>` и `<CFPOP>`, которые предназначены для отправления и получения почты соответственно.

Тег `<CFMAIL>`

Синтаксис:

```
<CFMAIL TO = "recipient"
      FROM = "sender"
      CC = "copy_to"
      BCC = "blind_copy_to"
```

```
SUBJECT = "msg_subject"
TYPE = "msg_type"
MAXROWS = "max_msgs"
MIMEATTACH = "path"
QUERY = "query_name"
GROUP = "query_column"
GROUPCASESENSITIVE = "Yes | No"
STARTROW = "query_row"
SERVER = "servername"
PORT = "port_ID"
MAILERID = "headerid"
TIMEOUT = "seconds">
```

```
    Содержание письма
</CFMAIL>
```

В табл. 16.1 представлено описание атрибутов тега <CFMAIL>.

Таблица 16.1. Описание атрибутов тега <CFMAIL>

Атрибут	Описание
TO	Электронный адрес получателя. Обязательный атрибут
FROM	Электронный адрес отправителя. Обязательный атрибут
CC	Электронный адрес для копии отправляемого письма. Не-обязательный атрибут ¹
BCC	Электронный адрес для копии отправляемого письма без внесения в список писем. Необязательный атрибут ²
SUBJECT	Тема письма. Обязательный атрибут
TYPE	Тип письма. В этой связи может быть использовано значение HTML. Этот атрибут полезен в случае отправления письма в HTML-формате. Необязательный атрибут
MAXROWS	Максимальное количество отправляемых писем. Используется вместе с атрибутом QUERY. Необязательный атрибут
MIMEATTACH	Файл, прилагаемый к письму, с указанием полного пути. Необязательный атрибут
QUERY	Наименование запроса. Используется для рассылки нескольких писем или для отправления результата запроса. Необязательный атрибут

¹ От англ. "Carbon Copy" — точная копия.
² От англ. "Blind Carbon Copy" — слепая точная копия.

Таблица 16.1 (окончание)

Атрибут	Описание
GROUP	Наименование колонки запроса. Используется вместе с атрибутом <code>QUERY</code> для группировки отправляемых писем по заданной колонке запроса. Необязательный атрибут
GROUPCASESENSITIVE	Восприимчивость к регистру. Используется для указания учета регистра при группировке результата запроса. Возможные значения: • Yes — да (по умолчанию); • No — нет. Необязательный атрибут
STARTROW	Номер записи, с которой следует посылать письма. Указывается вместе с атрибутом <code>QUERY</code> . Необязательный атрибут
SERVER	Адрес SMTP-сервера, используемый для рассылки писем. Если данный атрибут не задан, используется значение, указанное на странице "ColdFusion Administrator". Обязательный атрибут
PORT	TCP/IP-порт, используемый SMTP-сервером. Обычно принимает значение 25. Необязательный атрибут
MAILERID	Идентификатор отправителя, в заголовке получаемого письма значится как X-mailer. Необязательный атрибут
TIMEOUT	Период ожидания для соединения с SMTP-сервером. Измеряется в секундах. Необязательный атрибут

Рассылка почты

Теперь, когда у вас уже есть представление о возможных атрибутах тега `<CFMAIL>`, позволяющего отправлять почтовые сообщения, можно перейти к примерам. И в качестве одного из первых примеров приведем код, демонстрирующий отправление электронного письма, используя обязательные атрибуты:

```
<CFMAIL TO="Anna@MyMail"
      FROM="Ruben@MyMail"
      SUBJECT="Hello Anjuta"
      SERVER="192.168.3.68">
```

This message was sent by an

```
automatic mailer built with CFMAIL
</CFMAIL>
```

где в качестве SMTP-сервера указан IP-адрес компьютера с предустановленным специализированным программным продуктом. Впрочем, при реализации приложения можно использовать заранее объявленные переменные, как показано в следующем примере:

```
<HTML>
<HEAD>
  <TITLE>MyCFMail.cfm</TITLE>
</HEAD>
<BODY>
  <CFIF isDefined("Form.To")>
    <CFMAIL TO = #Form.To#
      FROM = #Form.From#
      SUBJECT = #Form.Subject#
      SERVER = #Application.smtp#>
    #Form.Body#
  </CFMAIL>
<CFELSE>
  <FORM action="MyCFMail.cfm" method="post" name="form1">
    <TABLE width="400" border="0" cellspacing="4" cellpadding="0">
      <TR><TD align="right">From</td>
        <TD><input name="From" size="20"></TD></TR>
      <TR><TD align="right">To</td>
        <TD><input name="To" size="20"></TD></TR>
      <TR><TD align="right">Subject</td>
        <TD><input name="Subject" size="40"></TD></TR>
      <TR><TD colspan="2">
        <textarea name="Body" rows="10" cols="70"></textarea>
      </TD></TR>
      <TR><TD align="right"><input type="submit" value="Send"></TD>
        <TD><input type="reset" value="Clear">
      </TD></TR>
    </TABLE>
  </FORM>
</CFIF>
</BODY>
</HTML>
```

Здесь переменная приложения `Application.smtp` объявляется в конфигурационном файле приложения `Application.cfm`, остальные же значения передаются из заранее подготовленной формы, отраженной в этом же примере.

Если же следует отправлять письма в виде HTML, то необходимо использовать атрибут `TYPE`, как показано ниже:

```
<CFMAIL TO="Anna@MyMail"
        FROM="Ruben@MyMail"
        SUBJECT="Hello Anjuta"
        TYPE="HTML"
        SERVER="192.168.3.68">

<h1>Hello Anjuta</h1>

<FONT color="Maroon" face="Bookman Old Style">
    How are you?
</FONT>

</CFMAIL>
```

Как видно из описания тега `<CFMAIL>`, для вложения файла в отправляемое письмо необходимо использовать атрибут `MIMEATTACH`. Однако вложение файлов также можно осуществлять, используя тег `<CFMAILPARAM>`, который в свою очередь должен быть помещен внутрь тега `<CFMAIL>`. Причем для вложения файла с использованием тега `<CFMAILPARAM>` необходимо указывать атрибут `FILE`, как показано в следующем примере:

```
<CFMAIL TO="Anna@MyMail"
        FROM="Ruben@MyMail"
        SUBJECT="Hello Anjuta"
        SERVER="192.168.3.68"
        MIMEATTACH="c:\temp\file1.doc">

Hello Anjuta

Send you three files.

<CFMAILPARAM FILE="c:\temp\file2.doc">
<CFMAILPARAM FILE="c:\temp\file3.htm">

</CFMAIL>
```

Здесь к письму прикладывается три файла, вложение двух из которых осуществляется посредством тега `<CFMAILPARAM>`, который можно использовать в одном письме более одного раза. Тег `<CFMAILPARAM>` также можно применять для встраивания в заголовок письма дополнительных параметров, как показано ниже:

```
<CFMAIL TO="Anna@MyMail"
        FROM="Ruben@MyMail"
        SUBJECT="Hello Anjuta"
```



```
SERVER="192.168.3.68">
Hello Anjuta
<CFMAILPARAM NAME="Gratitude" VALUE="For Anjuta">
Thank you for all.
</CFMAIL>
```

где с помощью атрибута NAME в заголовок письма был включен параметр с именем "Gratitude", а в качестве значения этого параметра было задано значение "For Anjuta" с помощью атрибута VALUE тега <CFMAILPARAM>.

Следующий пример демонстрирует возможность рассылки писем с использованием базы данных:

```
<HTML>
<HEAD>
  <TITLE>Seasonal dispatch</TITLE>
</HEAD>
<BODY>
  <CFQUERY NAME="GetTraveler" DATASOURCE="#application.dsn#">
    SELECT *
    FROM Traveler
    ORDER BY FirstName_tra, LastName_tra
  </CFQUERY>
  <CFMAIL TO = #Email_tra#
    FROM="Root@MyMail"
    SUBJECT="Change of the schedule"
    SERVER="192.168.3.68"
    QUERY="GetTraveler">
    Dear #FirstName_tra#
    The schedule of cruises has been changed
    due to the coming of the new season.
  </CFMAIL>
</BODY>
</HTML>
```

В приведенном выше примере предварительно был сформирован запрос с именем "GetTraveler", в последствии указанный в теге <CFMAIL>. В результате чего всем зарегистрированным любителям путешествовать будет отправлено уведомление об изменении расписания круизов в связи с переходом на новый сезон.

Ограничение же по рассылке уведомлений можно устанавливать как изменением непосредственно формы запроса, добавляя необходимые критерии,

так и с использованием необязательного атрибута `MAXROWS`, позволяющего установить максимальное количество отправляемых писем независимо от результата запроса.

Тег <CFPOP>

Синтаксис:

```
<CFPOP SERVER = "servername"
      PORT = "port_number"
      USERNAME = "username"
      PASSWORD = "password"
      ACTION = "GetHeaderOnly | GetAll | Delete"
      NAME = "queryname"
      MESSAGENUMBER = "number"
      ATTACHMENTPATH = "path"
      TIMEOUT = "seconds"
      MAXROWS = "number"
      STARTROW = "number"
      GENERATEUNIQUEFILENAMES = "Yes | No">
```

В табл. 16.2 представлено описание атрибутов тега <CFPOP>.

Таблица 16.2. Описание атрибутов тега <CFPOP>

Атрибут	Описание
SERVER	Адрес POP-сервера, используемый для получения писем. Обязательный атрибут
PORT	Порт, используемый POP-сервером. Обычно принимает значение 110. Необязательный атрибут
USERNAME	Учетная запись пользователя. При отсутствии указания пользователя связь приобретает анонимный характер (anonymous). Необязательный атрибут
PASSWORD	Пароль пользователя. Необязательный атрибут
ACTION	Определение действия по отношению к входящим письмам. Возможные значения: - GetHeaderOnly — возвращает частичную информацию, не включая вложенные файлы, содержание и подробный заголовок входящего письма (по умолчанию); - GetAll — возвращает всю информацию; - Delete — удаляет входящее письмо на POP-сервере

Таблица 16.2 (окончание)

Атрибут	Описание
NAME	Имя формирующегося запроса на основании полученных писем. Обязательный атрибут для таких действий, как <code>GetHeaderOnly</code> и <code>GetAll</code>
MESSAGENUMBER	Порядковый номер входящего письма. Обязательный атрибут для действия удаления <code>Delete</code> . Также можно использовать для действий <code>GetHeaderOnly</code> и <code>GetAll</code> в целях получения только определенных писем. Причем в качестве порядкового номера можно указывать несколько писем через запятую. Несуществующие письма игнорируются
ATTACHMENTPATH	Месторасположение вложенных файлов. Может использоваться в сочетании с действием <code>GetAll</code> . Необязательный атрибут
TIMEOUT	Период ожидания для обработки писем. По умолчанию 60 секунд. Необязательный атрибут
MAXROWS	Максимальное количество входящих писем. Игнорируется в случае использования атрибута <code>MESSAGENUMBER</code> . Необязательный атрибут
STARTROW	Номер записи, с которой следует обрабатывать письма. По умолчанию равен 1. Игнорируется в случае использования атрибута <code>MESSAGENUMBER</code> . Необязательный атрибут
GENERATEUNIQUEFILE NAMES	Индикатор, указывающий на создание уникальных имен для вложенных файлов, имеющих одинаковые имена с уже существующими размещенными файлами. Возможные значения: • Yes — да; • No — нет (по умолчанию). Необязательный атрибут

При использовании тега `<CFPOP>` создается объект запроса, обеспечивающий доступ к вспомогательной информации с помощью следующих переменных:

- ☐ `queryName.recordCount` — количество записей (писем), полученных в результате запроса;
- ☐ `queryName.currentRow` — текущая запись (письмо) запроса, при использовании тега `<CFOUTPUT>`;
- ☐ `queryName.columnList` — список колонок запроса, перечисленных через запятую.

При определении действия `GetHeaderOnly` в объект запроса включаются следующие колонки:

- ❑ `queryName.to` — электронный адрес получателя;
- ❑ `queryName.from` — электронный адрес отправителя;
- ❑ `queryName.cc` — электронный адрес для копирования входящего письма;
- ❑ `queryName.subject` — тема входящего письма;
- ❑ `queryName.date` — дата размещения в почтовом ящике;
- ❑ `queryName.messageNumber` — порядковый номер входящего письма;
- ❑ `queryName.replyTo` — содержание ответа.

При определении действия `GetAll` помимо вышеперечисленных колонок в объект запроса включаются также такие колонки, как:

- ❑ `queryName.body` — содержание входящего письма;
- ❑ `queryName.header` — подробный заголовок входящего письма;
- ❑ `queryName.attachments` — список вложенных файлов;
- ❑ `queryName.attachmentFiles` — список путей вложенных файлов.

Список как вложенных файлов, так и их путей перечисляется через восьмибитовый расширенный ASCII-код `CHR(9)` (знак табуляции).

Получение почты

Предположим, что нам необходимо вывести основные данные по полученным электронным сообщениям для конкретного пользователя с предварительным отражением количества принятых писем на POP-сервере:

```
<HTML>
<HEAD>
  <TITLE>CFPOP</TITLE>
</HEAD>
<BODY>
  <CFPOP SERVER = "192.168.3.68"
    USERNAME = "Anna"
    PASSWORD = "*****"
    NAME = "GetMail"
    ACTION="GetHeaderOnly">
    <h3>Your Inbox: <CFOUTPUT>#GetMail.RecordCount#</CFOUTPUT></h3>
    <TABLE border="1" cellpadding="2" cellspacing="2">
      <TR>
```

```

        <TD>Row</TD><TD>To</TD><TD>From</TD><TD>SUBJECT</TD>
        <TD>Date</TD>
    </TR>
    <CFOUTPUT QUERY="GetMail">
    <TR>
        <TD>#CurrentRow#</TD><TD>#To#</TD><TD>#From#</TD>
        <TD>#Subject#</TD>
        <TD>#DateFormat (ParseDateTime (Date) , "dd/mm/yyyy") #</TD>
    </TR>
    </CFOUTPUT>
</TABLE>
</BODY>
</HTML>

```

Здесь для вывода такой информации, как текущая запись, электронный адрес получателя, электронный адрес отправителя, тема входящего письма и дата размещения в почтовом ящике используется уже знакомый тег `<CFOUTPUT>` с атрибутом `QUERY`, указывающим на имя запроса. Заметим, что для преобразования текстового отображения даты в объект формата "дата/время" следует использовать функцию `ParseDateTime`, как показано в приведенном примере.

Достаточно часто вместе с почтовыми сообщениями передаются различные файлы, такие файлы мы называем *вложенными* или *прикрепленными* к письму. Причем к одному входящему почтовому сообщению может быть прикреплено сразу несколько файлов. В свою очередь тег `<CFPOP>` располагает таким атрибутом, как `ATTACHMENTPATH`, позволяющим устанавливать месторасположение вложенных файлов, а переменная `attachmentFiles` получаемого запроса в результате использования тега `<CFPOP>` принимает значение списка путей вложенных файлов. В этой связи в качестве следующего примера продемонстрируем возможность считывания вложенных файлов, не забывая, что для успешного использования переменной `attachmentFiles` необходимо указывать действие `ACTION`, равное `"GetAll"`:

```

<CFPOP SERVER = "192.168.3.68"
    USERNAME = "Anna"
    PASSWORD = "*****"
    NAME = "GetMailAttach"
    MESSAGENUMBER="2"
    ATTACHMENTPATH="c:\temp\my"
    ACTION="GetAll">
<CFOUTPUT QUERY="GetMailAttach">
    <TABLE border="1" width="400" cellpadding="2" cellspacing="2">

```

```

<TR><TD align="right"><b>From:</b></TD><TD>#From#</TD></TR>
<TR><TD align="right"><b>Subject:</b></TD><TD>#Subject#</TD></TR>
<TR><TD colspan="2">#Body#</TD></TR>
<TR><TD align="right" valign="top"><b>Attachments:</b></TD>
<TD>
    <CFLOOP INDEX="i" LIST=#AttachmentFiles# DELIMITERS=#Chr(9)#>
        <a href="file://#i#">#i#</a><br>
    </CFLOOP>
</TD></TR>
</TABLE>
</CFOUTPUT>

```

Здесь атрибут `MESSAGENUMBER` тега `<CFPOP>` указывает на порядковый номер входящего письма, которое в последствии является объектом для считывания вложенных файлов.

Еще раз отметим, что список путей вложенных файлов перечисляется через восьмибитовый расширенный ASCII-код `CHR(9)`, используемый в нашем случае в качестве разделителя при формировании списка прикрепленных файлов с помощью тега `<CFLOOP>`.

Для удаления входящих на POP-сервер писем достаточно указать тег `<CFPOP>` в сочетании с определением действия `ACTION`, равным `"Delete"`, и указанием порядкового номера письма с помощью атрибута `MESSAGENUMBER`. Для удаления же сразу нескольких писем можно использовать цикл или в качестве порядкового номера указывать список писем через запятую. Следующий пример демонстрирует удаление входящего письма с порядковым номером 1.

```

<CFPOP SERVER = "192.168.3.68"
    USERNAME = "Anna"
    PASSWORD = "*****"
    MESSAGENUMBER="1"
    ACTION="Delete"
    NAME= "DelMail">

```

Так же, как и с тегом `<CFMAIL>`, при обработке входящей почты можно применять заранее объявленные переменные, как показано в следующем примере:

```

<CFPOP SERVER = #ServerPOP#
    USERNAME = #Form.UserName#
    PASSWORD = #Form.Password#
    MESSAGENUMBER = #Form.MessNum#
    ACTION = "Delete"
    NAME="DelMail">

```

Пример реализации почтового ящика

В качестве примера рассмотрим создание собственной почтовой службы на Web-сервере, заранее установив один из почтовых серверов с именем "MyMail".

В первую очередь хотелось бы сформулировать конечную цель, что немало важно при реализации любого проекта. А задача между тем тривиальна — реализуется возможность читать и отправлять почту с помощью Web-браузера. Далее обсудим детали.

Соответственно с любым приложением, предназначенным для использования различными группами людей, следует включить форму регистрации, которая бы позволила определять уникальный профиль того или иного пользователя. Также необходимо предусмотреть возможность ведения индивидуальной адресной книги каждого пользователя.

С чего начать?

Конечно, очень важно, чтобы система управления электронной почтой была простой и удобной в использовании. Эту задачу каждый программист выполняет по мере своих возможностей и накопленного опыта. Что же касается удобства, то исходя из опыта уже существующих подобных приложений, электронные письма желательно разбивать на тематические разделы или так называемые папки. В этом плане можно было бы создать гибкую систему по управлению папками с возможностью добавления новых и удаления существующих папок. Однако в нашем примере будем довольствоваться набором стандартных разделов. Тем более, что список папок станет храниться в таблице базы данных, и для добавления модуля по управлению папками необходимо будет всего лишь несколько изменить наш код и добавить пару CFML-страниц. Между тем, в качестве стандартных папок будем использовать:

- ❑ **Inbox** (Входящие) — в эту папку будет помещаться вся входящая почта;
- ❑ **Outbox** (Исходящие) — в эту папку будет помещаться вся исходящая почта;
- ❑ **Deleted** (Удаленные) — в эту папку будут помещаться удаленные письма;
- ❑ **Draft** (Черновик) — папка для хранения писем, приготовленных для отправления;
- ❑ **Junk** (Хлам) — папка для хранения по сути не столь уж важных писем.

На рис. 16.1 изображена схема взаимодействия модулей системы почтовой службы.

На приведенной схеме помимо модулей показаны выполняемые действия по отношению к базе данных в виде цилиндрических фигур, где:

- ❑ **Insert** — добавление данных;
- ❑ **Update** — обновление данных;
- ❑ **Delete** — удаление данных.

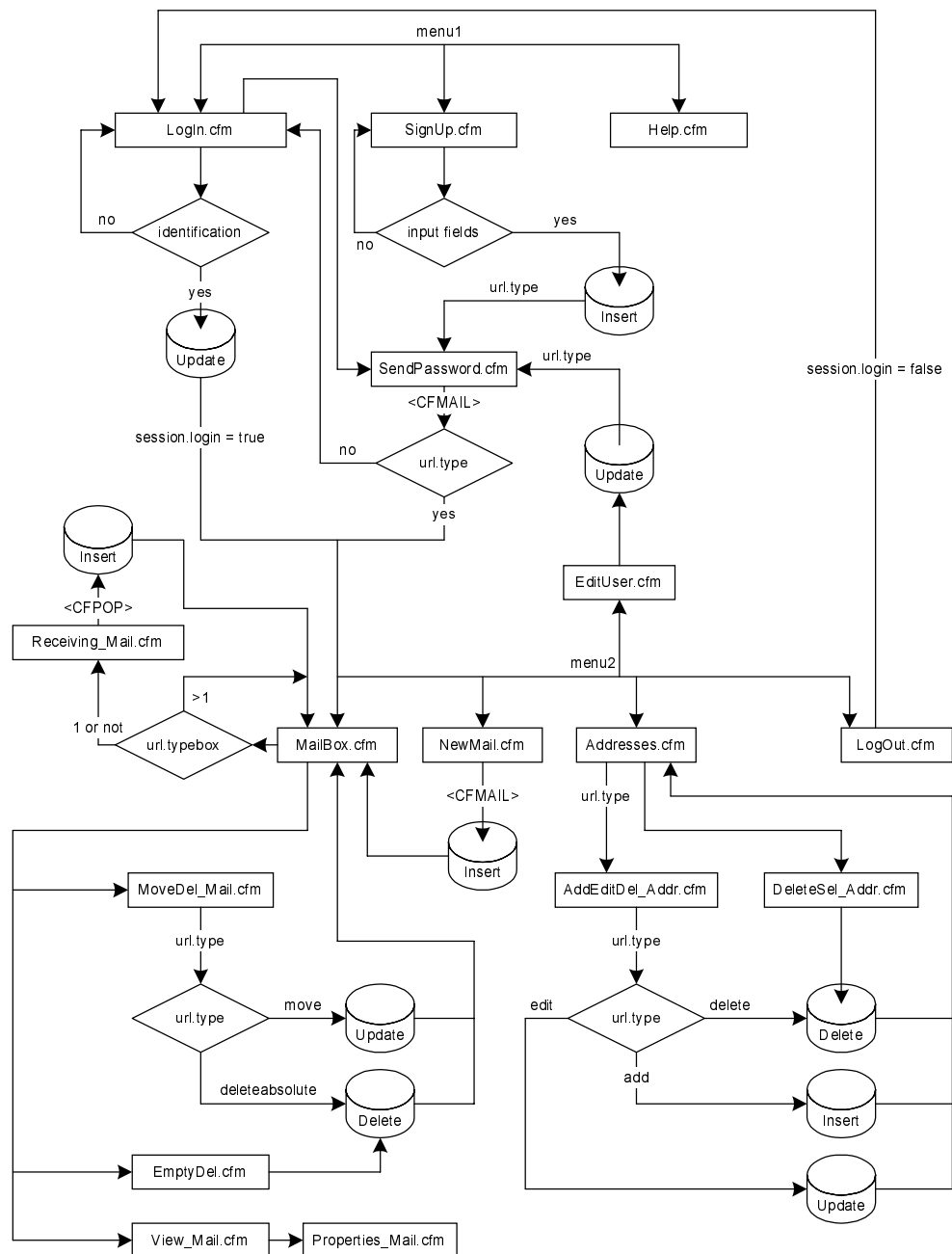


Рис. 16.1. Схема взаимодействия модулей системы почтовой службы

Для удобства просмотра такие действия, как `Select` (выборка данных) на приведенной принципиальной схеме не показаны.

Процесс же разработки базы данных подробно описан в *главе 5* на примере проекта "My-Line Cruises". Так что здесь мы покажем результат окончательной фазы проектирования базы данных `MyMail`, это так называемая физическая модель, для простоты реализованная под `Microsoft Access`. На рис. 16.2 показана физическая модель базы данных `MyMail`.

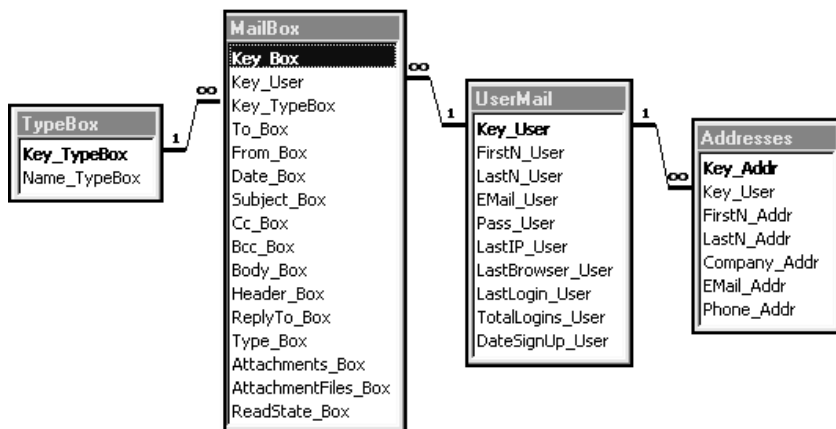


Рис. 16.2. Физическая модель базы данных `MyMail`

Регистрация

Теперь немного о том, как это работает. Изначально пользователю предлагается заполнить регистрационную карточку (`SignUp.cfm`), после чего при успешной регистрации на только что созданный почтовый ящик отправляется приветствие с уведомлением о внесенной информации, касающейся данного пользователя (`SendPassword.cfm`). Затем в окне браузера появляется новое меню с возможностью читать входящую почту (`MailBox.cfm`), создавать письма для последующей отправки (`NewMail.cfm`), формировать собственную адресную книгу (`Addresses.cfm`) и закрывать сессию (`Logout.cfm`) для предоставления возможных пауз во время работы с почтовой службой. При этом в дальнейшем для работы с данной службой достаточно будет только пройти идентификацию (`LogIn.cfm`), т. е. ввести свой электронный адрес и пароль, как показано на рис. 16.3.

Изменение же пароля возможно в любой момент после идентификации в режиме работы с программой `MyMail`, для чего достаточно всего лишь выбрать ссылку с собственным именем, находящуюся на текущей CFML-странице, и произвести необходимые изменения, как показано на рис. 16.4

(EditUser.cfm). При смене пароля система принудительно отправляет электронное сообщение о совершенных изменениях (SendPassword.cfm), что отображено на уже рассмотренной схеме (см. рис. 16.1).

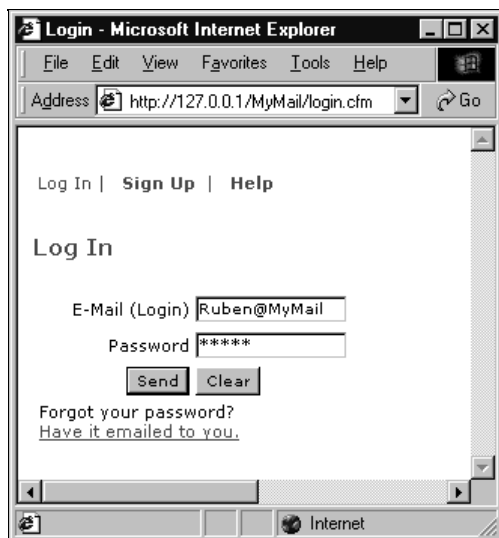


Рис. 16.3. Идентификация при доступе к почтовой службе на странице LogIn.cfm



Рис. 16.4. Изменение пароля пользователя почтовой службы на странице EditUser.cfm

Шифрование пользовательских паролей

В данном примере реализации почтового ящика мы впервые использовали так называемое *шифрование паролей*, хранящихся в базе данных. В ColdFusion существуют две функции: `Encrypt` и `Decrypt`. Первая из них — `Encrypt` — позволяет зашифровать текст по определенному алгоритму с помощью значения ключа. Вторая функция — `Decrypt` — соответственно позволяет расшифровать полученный результат при использовании того же ключа. Например, при регистрации пользователя можно совершить следующее действие:

```
<CFSET Password = Encrypt(Form.Pass_User, Session.user.id)>
```

где `Form.Pass_User` — введенный пользователем пароль; `Session.user.id` — уникальный ключ пользователя, присваиваемый на уровне триггера базы данных.

Следующим действием можно произвести обновление пароля для данного пользователя:

```
<CFQUERY NAME="UpdatePass" DATASOURCE="#Application.dsn#">
    UPDATE UserMail
    SET Pass_User = '#Password#'
    WHERE Key_User = #Session.user.id#
</CFQUERY>
```

Данный пример регистрации (`signup.cfm`) полностью представлен в листинге П2.7 приложения 2.

Теперь наш пароль хранится в базе данных в зашифрованном виде. Однако иногда возникает необходимость расшифровать пароль. Скажем, при требовании пользователя передать пароль по электронной почте или при идентификации пользователя в момент очередного посещения почтовой службы следует уже использовать функцию `Decrypt`. Например:

```
<CFIF isDefined("URL.Email")>
    <CFQUERY NAME="Verification" DATASOURCE="#Application.dsn#">
        SELECT Pass_User, FirstN_User, EMail_User, Key_User
        FROM UserMail
        WHERE EMail_User = '#URL.Email#'
    </CFQUERY>
    <CFIF Verification.RecordCount eq 1>
        <CFSET PostMasterEmail = "Root" & #Application.mailserver#>
        <CFSET Password = Decrypt(Verification.Pass_User,
            Verification.Key_User)>
```

```
<!-- Send EMail Password --->
<CFMAIL TO="#URL.Email#"
        FROM="#PostMasterEMail#"
        SUBJECT="Password"
        SERVER="#Application.smtp#">
    Dear #Verification.FirstN_User#
```

This message was sent at your request.

For work with the project "#Application.ApplicationName#",
you are registered E-Mail address(Login): #Verification.EMail_User#
with the password: #Password#

Best wishes.

```
</CFMAIL>
```

```
</CFIF>
```

```
<CFELSE>
```

```
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
```

```
</CFIF>
```

Здесь в первую очередь анализируется наличие пользователя с введенным электронным адресом, после чего при успешном результате поиска расшифровывается пароль этого пользователя с указанием необходимого ключа. А уж следующим шагом отправляется письмо по указанному адресу с вложением в текст письма результата расшифровки.

Данный пример отправления пароля (sendpassword.cfm) полностью представлен в листинге П2.8 *приложения 2*.

Если вести речь о шифровании паролей, то в ColdFusion также представлена функция Hash, позволяющая конвертировать текст в 32-битный код по одностороннему алгоритму MD5. Применение данной функции уже не позволит нам передавать пароли забывчивым пользователям, однако и не во всех системах это требуется. Например, можно поступить так:

```
<CFQUERY NAME="AddUser" DATASOURCE="#Application.dsn#">
    INSERT INTO UserMail(EMail_User, FirstN_User,
        LastN_User, Pass_User,
        DateSignUp_User, LastLogin_User,
        LastIP_User, LastBrowser_User, TotalLogins_User)
VALUES('#Email_User#', '#Form.FirstN_User#',
        '#Form.LastN_User#', '#Hash(Form.Pass_User)#',
        '#DateNow#', '#DateNow#',
```

```
'#CGI.REMOTE_ADDR#', '#CGI.HTTP_USER_AGENT#', 1)
```

```
</CFQUERY>
```

где вместо введенного пароля в базу данных благополучно записывается неподдающийся расшифровке результат функции `Hash`.

В данном случае уже при идентификации пользователя введенный им пароль необходимо конвертировать с помощью все той же функции `Hash` и затем полученный результат сравнивать со значением, находящимся в базе данных в качестве пароля. При этом идентификацию можно считать удачной, если сравниваемые значения окажутся идентичными.

Адресная книга

Что собой представляет адресная книга? При формировании тех или иных почтовых сообщений всегда присутствует понятие адресатов, т. е. "объектов внимания", которым, собственно говоря, вы и адресуете свои лирические послания и от которых могут приходить ответные сообщения. Более того, достаточно часто приходится отправлять электронные письма одним и тем же адресатам, и в этой связи с точки зрения пользователя было бы удобно выбирать электронный адрес получателя из существующего списка, вместо того, чтобы каждый раз набирать его адрес заново. Для решения данной задачи многие разработчики почтовых систем реализуют адресную книгу, позволяющую пользователям формировать список часто используемых электронных адресов. Причем в соответствии с электронным адресом следует хранить дополнительные данные, позволяющие охарактеризовать тот или иной вновь добавленный адресат или, можно назвать иначе, контакт. Создавая сложную по структуре адресную книгу, необходимо предусмотреть возможность объединения контактов в определенные группы, мы же облегчим себе задачу, реализуя простую адресную книгу, куда будут включены помимо электронного адреса (e-mail) такие поля, как:

- ☐ First Name — имя;
- ☐ Last Name — фамилия;
- ☐ Company — компания;
- ☐ Phone — телефон.

На рис. 16.5 продемонстрирован пример адресной книги с заранее заполненными данными.

В связи с тем, что адресная книга может разрастись до "немыслимых" размеров, Web-страница, отображающая список контактов, разделена на блоки. В нашем случае каждый блок содержит по семь контактов, однако данное значение можно регулировать. Для этой цели в конфигурационный файл (`Application.cfm`) данного приложения включена переменная `Application.PageRows`. Также в конфигурационный файл приложения добавлена переменная `Application.MaxLinePages`, позволяющая регулировать

количество данных блоков, располагаемых на одной линии. Изначально это значение равняется 12. Более подробно о настройке конфигурационных файлов читайте в *главе 10*.

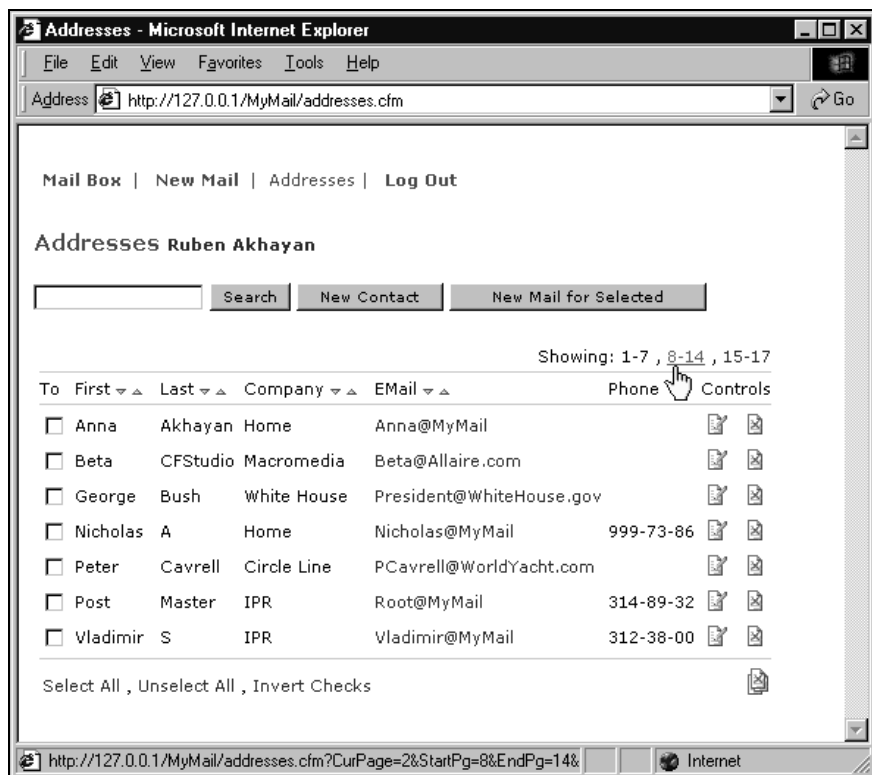


Рис. 16.5. Пример адресной книги Addresses.cfm

В разработанной нами адресной книге имеется дополнительная колонка под названием **Controls** (Управление), где напротив каждого контакта располагаются два элемента в виде графических изображений: **Edit** (Редактировать) и **Delete** (Удалить), позволяющие соответственно редактировать и удалять данный контакт. В нижней же части управляющей колонки находится элемент **Delete Selected** (Удалить выделенные), предназначенный для группового удаления контактов, предварительно выделенных с помощью вспомогательных элементов **Select All** (Выделить все), **Unselect All** (Отмена выделения для всех) или **Invert Checks** (Обратить отмеченные), расположенных в нижней части Web-страницы. Для добавления нового контакта достаточно нажать кнопку **New Contact** (Новый контакт) и заполнить предлагаемые поля.

Каждый электронный адрес контакта, расположенный в колонке **EMail** (e-mail), является ссылкой, при щелчке по которой открывается страница

NewMail.cfm, предназначенная для создания нового почтового сообщения с автоматически заполненным полем **To** (Кому), где значением поля является выбранный электронный адрес. Если же существует необходимость создать почтовое сообщение сразу для нескольких адресатов, достаточно, отметив таковые либо вручную, либо при помощи все тех же вспомогательных инструментов группового выделения, нажать кнопку **New Mail for Selected** (Новое сообщение для выделенных контактов).

Как видно из рис. 16.5, в заголовке практически каждой колонки помимо названия находятся два элемента в виде стрелочек, повернутых в разные стороны: вниз и вверх. Так вот эти инструменты предполагается использовать для сортировки списка контактов как по возрастанию, так и по убыванию в соответствии с направлением стрелок. Причем изначально список отсортирован по возрастанию первого имени (**First**) адресата.

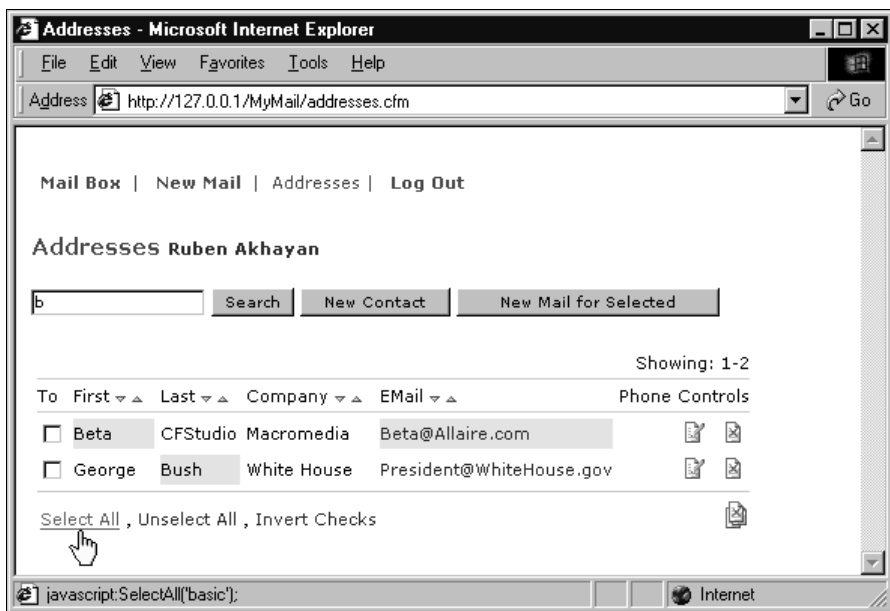


Рис. 16.6. Результат поиска в адресной книге

Мы уже говорили о возможно большом количестве контактов, для чего данный список разделяется на страничные блоки. Впрочем, иногда пользователю требуется найти конкретный адрес по ключевому слову, не прибегая к поблочному перелистыванию. И для этой цели мы включили два элемента: поле для ввода искомых фрагментов текста и кнопку **Search** (Поиск) для выполнения поиска. К примеру, введем в поле ввода символ **b** и выполним поиск. На рис. 16.6 продемонстрирован результат поиска.

Как видно из рисунка, были найдены две записи, где цветом выделены поля, включающие искомый фрагмент текста. Обратите внимание, что при поиске игнорируется разница между верхним и нижним регистрами.

Создание и отправление почтовых сообщений

Скажем несколько слов о создании почтовых сообщений. Уже было отмечено, что в нашем примере за создание писем отвечает Web-страница NewMail.cfm. На этой странице представлена форма для заполнения необходимых данных, подкрепленная ссылками для полей, требующих ввод электронного адреса: поле **То** (Кому), поле **Сс** (Точная копия) и поле **Всс** (Слепая точная копия). Эти ссылки предусмотрены для вызова списка электронных адресов из заранее сформированной адресной книги, о которой уже достаточно много было сказано.

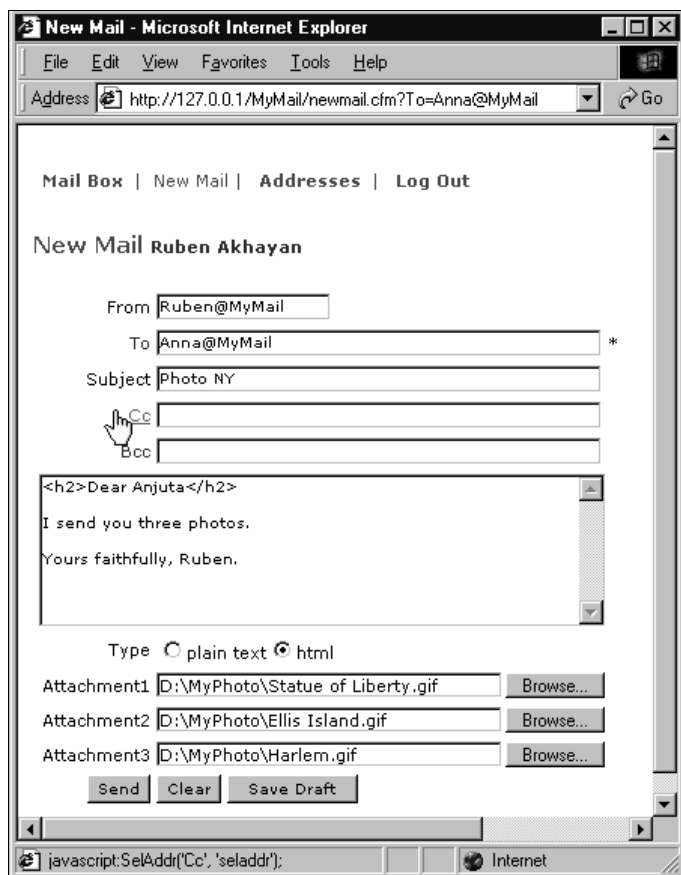


Рис. 16.7. Пример составления электронного сообщения NewMail.cfm

В режиме составления писем можно устанавливать тип письма (**Type**) либо в виде простого текста (переключатель **plain text**), либо в виде HTML (переключатель **html**). Также в нашей почтовой службе предусмотрена возможность прикрепления к письму до трех файлов различного формата. На рис. 16.7 продемонстрирован пример составления электронного сообщения.

Заметим, что помимо отправления письма с помощью нажатия кнопки **Send** (Отправить) есть возможность сохранить составленное письмо в разделе черновиков, используя при этом кнопку **Save Draft** (Сохранить как черновик). Таким образом, сохраненное письмо в виде черновика в любой момент можно предварительно просмотреть, отредактировать или отправить по указанному адресу.

Просмотр папок с письмами

Если лицо, являющееся получателем составленного нами письма, также пользуется данной почтовой службой, то при входе в систему он или она, к примеру, может наблюдать следующую картину, представленную на рис. 16.8.

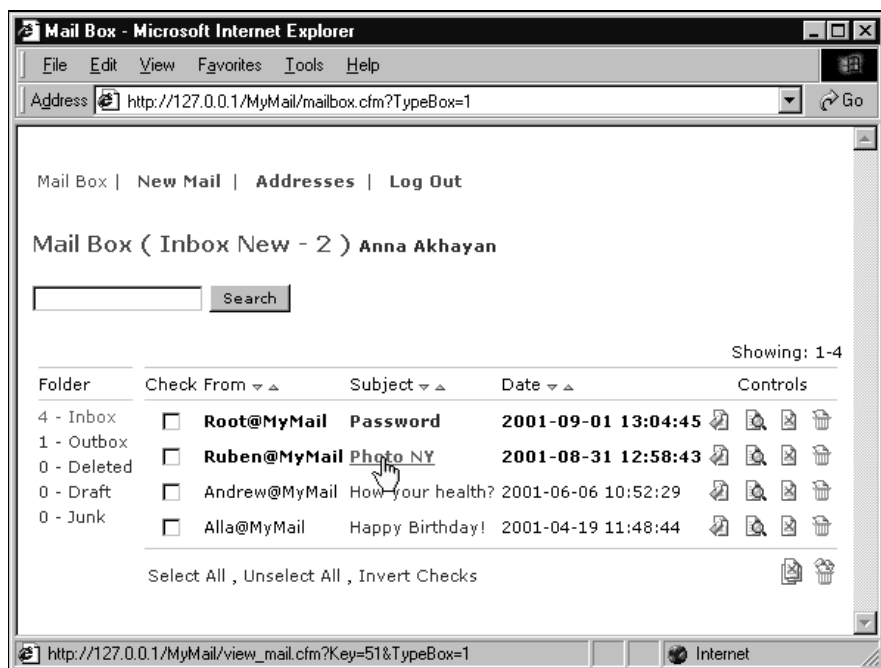


Рис. 16.8. Пример отображения входящих писем MailBox.cfm

Важно подчеркнуть, что в этом случае количество новых писем складывается из входящих с POP-сервера и непрочитанных, но хранящихся в базе дан-

ных. Причем, как только система обнаруживает новые письма на POP-сервере, эти письма тут же добавляются в базу данных со свойством "не прочитано", и затем POP-сервер освобождается от них. Непрочитанные же письма выделяются жирным шрифтом.

Как видно из рис. 16.8, для писем так же, как и для адресной книги, предусмотрен инструмент поиска, действующий по той же схеме. Также здесь имеется колонка управления **Controls** (Управление), составленная из следующих элементов в виде графических изображений:

- ☐ **Reply** (Ответить);
- ☐ **View** (Просмотреть);
- ☐ **Delete** (Удалить);
- ☐ **Move junk** (Переместить в папку Хлам).



Рис. 16.9. Пример просмотра писем View_Mail.cfm

При всем при том, элемент **Delete** (Удалить) письма, находящиеся в папках **Inbox** (Входящие) и **Junk** (Хлам), перемещает в папку **Deleted** (Удаленные).

По отношению к другим папкам элемент **Delete** (Удалить) безвозвратно удаляет письма из базы данных.

На рис. 16.9 представлен пример просмотра отправленного нами письма.

Для удобства здесь же имеется возможность вернуться обратно к списку входящих писем. Используя элементы перемещения от письма к письму в виде четырех разнонаправленных стрелок, расположенных в правом верхнем углу текущей Web-страницы, можно просматривать другие письма. Также здесь дублируются элементы управления, перечисленные выше. К примеру, если после просмотра полученного письма вы решили немедленно отправить ответ на него, то это можно сделать при помощи элемента **Reply** (Ответить), аргументом для которого будет являться уникальный код письма.

Что же касается вложенных файлов, то они располагаются списком в виде ссылок, позволяющих открывать данные файлы.

И последнее в отношении просмотра писем: любое входящее письмо имеет подробный заголовок, не всегда желанный для чтения. И по этой причине на странице просмотра мы добавили кнопку **Properties** (Свойства), позволяющую открывать текст подробного заголовка в отдельном окне. При этом за содержание свойств текущего письма отвечает страница `Properties_Mail.cfm`. Данный заголовок мы также используем для анализа типа входящих писем. При существовании в заголовке входящего письма фрагмента "Content-type: text/html" данное письмо помещается в базу данных с отметкой "HTML", что позволяет регулировать вид просмотра при чтении писем. К примеру:

```
<CFQUERY NAME="SaveMail" DATASOURCE="#Application.dsn#">
  <CFIF FindNoCase('Content-type: text/html', Header) gt 0>
    <CFSET conType = "HTML">
  <CFELSE>
    <CFSET conType = "">
  </CFIF>

  INSERT INTO MailBox(Key_User, Key_TypeBox, To_Box, From_Box,
    Cc_Box, Subject_Box, Body_Box, ReplyTo_Box,
    Type_Box, Date_Box, ReadState_Box, Header_Box,
    Attachments_Box, AttachmentFiles_Box)
  VALUES(#Session.user.id#, 1, '#To#', '#From#',
    '#Cc#', '#Subject#', '#Body#', '#Replyto#',
    '#conType#', '#Date_#', 0, '#Header#',
    '#Attachments#', '#Attachmentfiles#')
</CFQUERY>
```

На рис. 16.10 показан пример списка исходящих писем.



Рис. 16.10. Пример отображения исходящих писем

Список исходящих писем только тем и отличается от списка входящих, что для исходящих реализована колонка **To** (Кому), а для входящих писем отображается колонка **From** (От кого). И как уже было отмечено, при удалении входящей почты письма перемещаются в папку **Deleted** (Удаленные), а подобное действие, совершаемое по отношению к исходящей почте, приводит к удалению писем из базы данных.

На рис. 16.11 показан пример списка удаленных писем.

Для почты, расположенной в папке удаленных писем, характерно, что данные письма всегда можно восстановить. И для этой цели добавлен управляющий элемент **Undo** (Восстановить). Если, конечно, перед этим не была выполнена команда очистки писем, помеченных на удаление — **Empty Deleted** (Опустошить помеченные на удаление).

На рис. 16.12 показан пример списка писем, помещенных в папку черновиков.

Письма, находящиеся в папке **Draft** (Черновик), по сути можно рассматривать как сформированные, но отложенные для отправки на некоторый неопределенный срок. Это достаточно удобно, когда следует заранее написать ряд писем и отправить их только через некоторое время. При этом, имея возможность пересмотреть содержимое данных писем, либо вообще отказаться от отправления таковых.

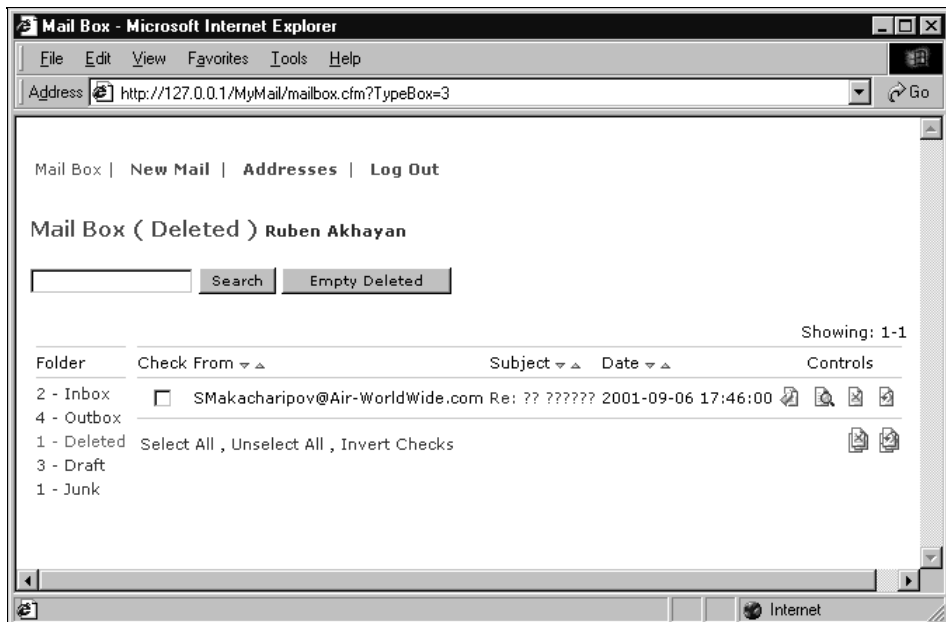


Рис. 16.11. Пример отображения удаленных писем

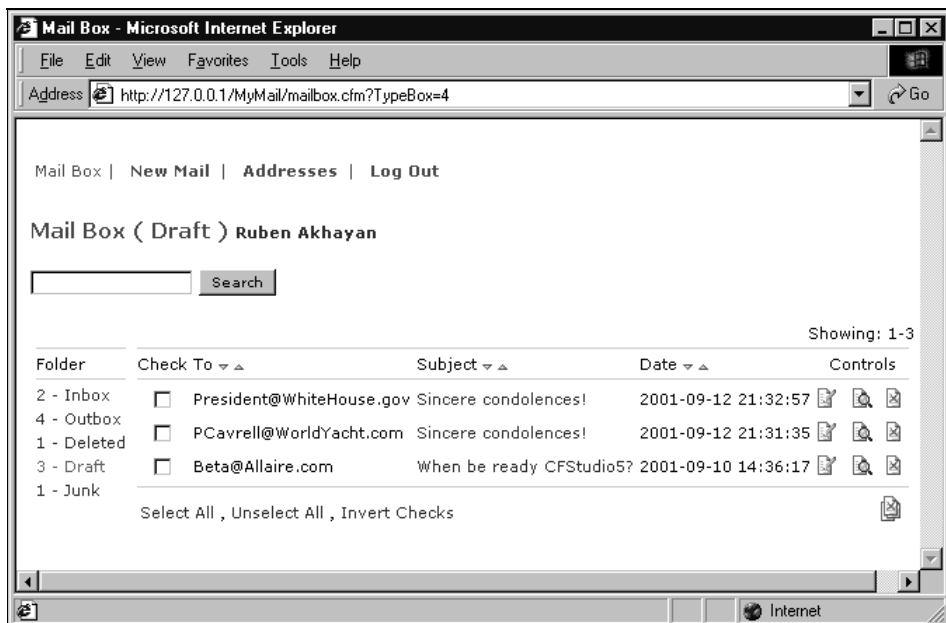


Рис. 16.12. Пример отображения писем, помещенных в папку черновиков

На рис. 16.13 показан пример списка писем, помещенных в папку ненужных сообщений.

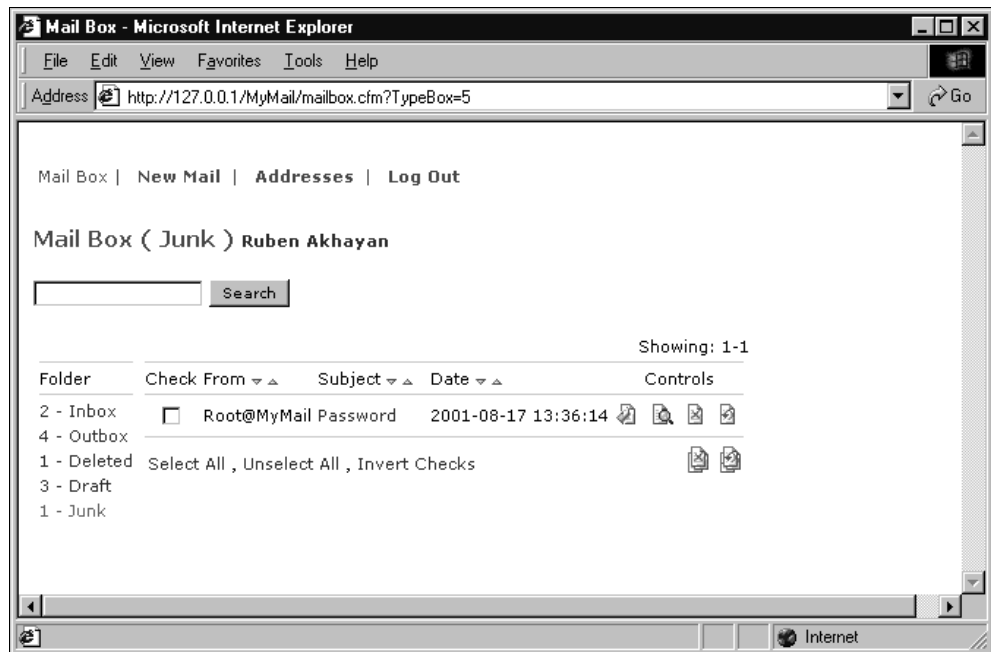


Рис. 16.13. Пример отображения писем помещенных, в папку ненужных сообщений

Письма, находящиеся в папке **Junk** (Хлам), так же, как и письма, помеченные на удаление, можно восстановить. То есть все восстановленные письма будут перемещены обратно в папку **Inbox** (Входящие).

Заключение по реализации почтового ящика

Конечно, можно было бы пойти дальше, предоставив потенциальному пользователю дополнительные возможности, такие как: управление папками, группировка контактных лиц, возможность обмениваться контактами, создание учетных записей для чтения писем из нескольких почтовых ящиков, проверка правописания, вставка ссылок, форматирование текста, возможность автоматической перекодировки, использование шаблонов, установка фильтров и т. д.

Однако представленный в этой главе пример реализации почтового ящика был создан в достаточно короткие сроки и не рассматривается как идеальный продукт. Целью же программирования данной системы является демонстрация возможностей ColdFusion Server.

В *приложении 2* представлены коды файлов в рамках проекта "MyMail".

Резюме

Подведем итог. В данной главе мы использовали CFML-теги `<CFMAIL>`, `<CFMAILPARAM>`, `<CFPOP>` и CFML-функции `ParseDateTime`, `Chr`, `Encrypt`, `Decrypt`, `Hash`.



Управление файлами и каталогами

При использовании любой операционной системы (ОС) вы, так или иначе, пользуетесь реализуемой с ее помощью возможностью организации хранения информации на вашем устройстве. Способы хранения могут быть различными, но смысл при этом не меняется — всегда существует *файловая система*.

В этой связи любая файловая система предполагает управление находящимися в этой системе файлами и каталогами. В ColdFusion существуют теги `<CFFILE>` и `<CFDIRECTORY>`, позволяющие управлять файловой системой на сервере. Причем тег `<CFFILE>` предназначен для выполнения определенных действий над файлами, а тег `<CFDIRECTORY>` в свою очередь можно использовать для управления каталогами. В этой главе будут подробно описаны возможные действия, совершаемые с помощью указанных тегов.

Тег `<CFFILE>`

Синтаксис тега `<CFFILE>` зависит от того действия, определяемого по отношению к файлу с помощью атрибута `ACTION`, который может принимать следующие значения:

- ☐ Upload — загрузка;
- ☐ Move — перемещение;
- ☐ Rename — переименование;
- ☐ Copy — копирование;
- ☐ Delete — удаление;
- ☐ Read — чтение;
- ☐ ReadBinary — чтение бинарных файлов;
- ☐ Write — запись;
- ☐ Append — добавление.

Например, для действия загрузки файла синтаксис тега выглядит так:

```
<CFFILE ACTION = "Upload"
    FILEFIELD = "formfield"
    DESTINATION = "full_path_name"
    NAMECONFLICT = "Error | Skip | Overwrite | MakeUnique"
    ACCEPT = "mime_type/file_type"
    MODE = "permission"
    ATTRIBUTES = "file_attributes">
```

Далее приведем таблицу возможных атрибутов в зависимости от используемого действия без учета уже описанного значения Upload (табл. 17.1).

Таблица 17.1. Список возможных атрибутов для разных действий тега <CFFILE>

Действие	Список возможных атрибутов
Move	SOURCE, DESTINATION, ATTRIBUTES
Rename	SOURCE, DESTINATION, ATTRIBUTES
Copy	SOURCE, DESTINATION, ATTRIBUTES
Delete	FILE
Read	FILE, VARIABLE
ReadBinary	FILE, VARIABLE
Write	FILE, OUTPUT, MODE, ADDNEWLINE, ATTRIBUTES
Append	FILE, OUTPUT, ATTRIBUTES

В табл. 17.2 представлено описание атрибутов тега <CFFILE>.

Таблица 17.2. Описание атрибутов тега <CFFILE>

Атрибут	Описание
ACTION	Действие, выполняемое над файлом (см. выше). Обязательный атрибут
FILEFIELD	Поле формы, передающее в качестве значения файл для загрузки. Обязательный атрибут только для действия Upload, не требует использования символов #, чтобы определить переменную
DESTINATION	Полный путь к каталогу или полное имя файла загрузки, перемещения, переименования или копирования. В сочетании с действием Rename следует указывать полное имя файла. Обязательный атрибут

Таблица 17.2 (продолжение)

Атрибут	Описание
NAMECONFLICT	Характеризует поведение при конфликтной ситуации, когда указанный файл уже существует. Возможные значения: • <code>Error</code> — файл не сохраняется, ColdFusion прекращает обрабатывать страницу и выдает ошибку (по умолчанию); • <code>Skip</code> — не сохраняет новую версию файла; • <code>Overwrite</code> — заменяет существующий файл; • <code>MakeUnique</code> — создается файл с уникальным именем. Необязательный атрибут
ACCEPT	Ограничение на тип файла. В качестве ограничения может быть указано несколько значений через запятую. Каждое значение состоит из двух составляющих "тип MIME/тип файла". Необязательный атрибут
MODE	Определяет доступ к файлу в ОС UNIX, Linux, Solaris, HP-UX. В Windows игнорируется. Необязательный атрибут
ATTRIBUTES	Определяет атрибуты файла через запятую. Возможные значения: • <code>ReadOnly</code> — только для чтения; • <code>Temporary</code> — временный; • <code>Archive</code> — архивный; • <code>Hidden</code> — скрытый; • <code>System</code> — системный; • <code>Normal</code> — нормальный. Необязательный атрибут
SOURCE	Полное имя файла, используемого в качестве источника того или иного действия. Только для действий <code>Move</code>, <code>Rename</code> и <code>Copy</code>. Обязательный атрибут
FILE	Полное имя файла. Обязательный атрибут. Только для действий <code>Delete</code>, <code>Read</code>, <code>ReadBinary</code>, <code>Write</code> и <code>Append</code>
VARIABLE	Название переменной, которая будет принимать значение содержания файла. Обязательный атрибут. Только для действий <code>Read</code> и <code>ReadBinary</code>
OUTPUT	Содержимое, записываемое в указанный файл. Только для действий <code>Write</code> и <code>Append</code>. Обязательный атрибут

Таблица 17.2 (окончание)

Атрибут	Описание
ADDNEWLINE	Добавление в конец текста символа перехода на новую строку. Только для действия Write. Возможные значения: • Yes — да (по умолчанию); • No — нет. Необязательный атрибут

Следует отметить, что при действии загрузки файла (Upload) с использованием тега <CFFILE> создается объект `cffile`, обеспечивающий доступ к вспомогательной информации с помощью следующих переменных:

- ☐ `cffile.attemptedServerFile` — имя указанного файла, принимает значение, если в качестве атрибута `DESTINATION` указано полное имя файла загрузки;
- ☐ `cffile.clientDirectory` — изначальное месторасположение загружаемого файла;
- ☐ `cffile.clientFile` — имя загружаемого файла, со стороны клиента;
- ☐ `cffile.clientFileExt` — расширение файла, загружаемого со стороны клиента;
- ☐ `cffile.clientFileName` — имя файла без расширения, загружаемого со стороны клиента;
- ☐ `cffile.contentSubType` — содержание типа сохраненного файла;
- ☐ `cffile.contentType` — содержание типа MIME сохраненного файла;
- ☐ `cffile.dateLastAccessed` — дата загрузки файла во время последнего обращения;
- ☐ `cffile.fileExisted` — определяет наличие файла по указанному пути и принимает значения Yes или No;
- ☐ `cffile.fileSize` — размер загружаемого файла;
- ☐ `cffile.fileWasAppended` — указывает, добавляется ли загружаемый файл в конец существующего, и принимает значения Yes или No;
- ☐ `cffile.fileWasOverwritten` — указывает, перезаписывается ли загружаемый файл поверх уже существующего, и принимает значения Yes или No;
- ☐ `cffile.fileWasRenamed` — указывает, переименовывается ли загружаемый файл в связи с ситуацией, когда файл уже существует, и принимает значения Yes или No;

- ❑ `cffile.fileWasSaved` — указывает, сохраняется ли загружаемый файл, и принимает значения `Yes` или `No`;
- ❑ `cffile.oldFileSize` — указывает размер файла, который был переписан, если таковой существует;
- ❑ `cffile.serverDirectory` — каталог на сервере, куда был сохранен загружаемый файл;
- ❑ `cffile.serverFile` — имя загружаемого файла, сохраненного на сервере;
- ❑ `cffile.serverFileExt` — расширение загружаемого файла, сохраненного на сервере;
- ❑ `cffile.serverFileName` — имя загружаемого файла без расширения, сохраненного на сервере;
- ❑ `cffile.timeCreated` — дата и время создания загружаемого файла;
- ❑ `cffile.timeLastModified` — дата и время последней модификации загружаемого файла.

Примеры управления файлами

Для понимания описанного выше синтаксиса тега, предназначенного для работы с файлами, приведем несколько примеров. Следующий пример позволяет нам записать определенный текст в указанный файл:

```
<CFFILE ACTION = "Write"
    FILE = "C:\Apache\htdocs\my-app\city.htm"
    OUTPUT = "<html><body><h1>NewYork</h1>">
```

Причем, если указанный файл не существует, то ColdFusion создает его в соответствии с определяемым месторасположением. Как видно из приведенного примера, в файле `city.htm` не были закрыты основные теги. Чтобы исправить данную ошибку, представим другой пример, позволяющий добавлять необходимый фрагмент текста:

```
<CFFILE ACTION = "Append"
    FILE = "C:\Apache\htdocs\my-app\city.htm"
    OUTPUT = "</body></html>">
```

Еще один пример, приведенный ниже, предполагает копирование существующего файла по указанному пути с атрибутом "только для чтения":

```
<CFFILE ACTION = "Copy"
    SOURCE = "C:\Apache\htdocs\my-app\city.htm"
    DESTINATION = "C:\InetPub\wwwroot\my-app\"
    ATTRIBUTES = "ReadOnly">
```

где в качестве атрибута `DESTINATION` может быть указано полное имя файла. Для удаления файла можно использовать следующий вариант:

```
<CFFILE ACTION = "Delete"
      FILE = "C:\Apache\htdocs\my-app\city.htm">
```

Представим пример для загрузки файлов с предварительно открывающейся диалоговой формой, позволяющей выбрать необходимый файл (листинг 17.1).

Листинг 17.1. Код страницы `TestUploadFile.cfm`

```
<HTML>
<HEAD>
  <TITLE>Upload</TITLE>
</HEAD>
<BODY>
  <h3>Upload File</h3>
  <CFIF isDefined("Form.FiletoUpload")>
    <CFFILE ACTION = "Upload"
          DESTINATION = "C:\InetPub\wwwroot\my-app\"
          NAMECONFLICT = "Overwrite"
          FILEFIELD = "Form.FiletoUpload">

  <CFOUTPUT>
    The information about upload a file<br>
    Name: #cfile.serverFile#<br>
    Source: #cfile.clientDirectory#<br>
    Size: #cfile.fileSize#<br>
    Last Modified: #cfile.timeLastModified#<br>
  </CFOUTPUT>
  <CFELSE>
    <FORM action="TestUploadFile.cfm"
          enctype="multipart/form-data" method="post">
      File for Upload:
      <input type="file" name="FiletoUpload" size="30"><br>
      <input type="submit" value="UpLoad">
    </FORM>
  </CFIF>
</BODY>
</HTML>
```

Здесь с помощью встроенных переменных при загрузке выбранного файла пользователю предоставляется информация об этом файле, в частности, имя файла (`cffile.serverFile`), исходное месторасположение загружаемого файла (`cffile.clientDirectory`), размер (`cffile.fileSize`) и дата и время последней модификации (`cffile.timeLastModified`).

Тег <CFDIRECTORY>

Как уже было отмечено, тег <CFDIRECTORY> позволяет управлять каталогами на сервере. Данный тег, так же, как и тег <CFFILE>, имеет атрибут ACTION, который может принимать следующие значения:

- ☐ List — список (по умолчанию);
- ☐ Create — создать;
- ☐ Delete — удалить;
- ☐ Rename — переименовать.

```
<CFDIRECTORY ACTION = "List | Create | Delete | Rename"
    DIRECTORY = "directory name"
    NAME = "query name"
    FILTER = "list filter"
    MODE = "permission"
    SORT = "sort specification"
    NEWDIRECTORY = "new directory name">
```

В табл. 17.3 представлено описание атрибутов тега <CFDIRECTORY>.

Таблица 17.3. Описание атрибутов тега <CFDIRECTORY>

Атрибут	Описание
ACTION	Действие, подробно описанное выше. Обязательный атрибут
DIRECTORY	Имя каталога, по отношению к которому предполагается совершение определенного действия. Обязательный атрибут
NAME	Имя запроса (query), для которого выбираются данные в виде списка параметров. Обязательный атрибут для действия List. Для других действий игнорируется
FILTER	Фильтр, позволяющий ограничивать полученный список. Не-обязательный атрибут. Следует использовать только с дейст-вием List. Можно применять специальные символы подста-новки * и ?
MODE	Определяет доступ к файлу в ОС UNIX, Linux, Solaris, HP-UX. В Windows игнорируется. Необязательный атрибут

Таблица 17.3 (окончание)

Атрибут	Описание
<code>SORT</code>	Значение сортировки списка. Необязательный атрибут. Следует использовать только с действием <code>List</code> . Можно указывать значение <code>ASC</code> (по возрастанию) и <code>DESC</code> (по убыванию), определяя вид сортировки. Параметры сортировки необходимо использовать через запятую. Например: <code>datelastmodified DESC, name ASC, size DESC</code>
<code>NEWDIRECTORY</code>	Имя нового каталога. Обязательный атрибут для действия <code>Rename</code> . Для других действий игнорируется

Отметим, что при действии получения списка (`List`) тега `<CFDIRECTORY>` создается объект запроса, обеспечивающий доступ к вспомогательной информации с помощью следующих переменных:

- `queryName.Name` — имя каталога или файла;
- `queryName.Size` — размер файла;
- `queryName.Type` — тип, принимающий значение `Dir` или `File`;
- `queryName.DateLastModified` — дата и время последней модификации;
- `queryName.Attributes` — атрибуты файла;
- `queryName.Mode` — доступ, в Windows игнорируется.

Примеры управления каталогами

И опять же несколько примеров. Для создания нового каталога достаточно выполнить код:

```
<CFDIRECTORY ACTION = "Create"
    DIRECTORY = "C:\Apache\htdocs\my-app\gif">
```

Переименовать только что созданный каталог можно следующим образом:

```
<CFDIRECTORY ACTION = "Rename"
    DIRECTORY = "C:\Apache\htdocs\my-app\gif"
    NEWDIRECTORY = "C:\Apache\htdocs\my-app\jpg">
```

Соответственно, для удаления каталога можно воспользоваться конструкцией:

```
<CFDIRECTORY ACTION = "Delete"
    DIRECTORY = "C:\Apache\htdocs\my-app\jpg">
```

И в завершение представим пример, позволяющий выводить список каталогов и файлов в соответствии с установленным текущим каталогом:

```
<CFDIRECTORY ACTION = "List"
    DIRECTORY = "C:\Apache\htdocs\my-app\"
    NAME = "QueryDir"
    FILTER = "cf*.*"
    SORT = "datelastmodified DESC, name ASC, size DESC">
<TABLE cellpadding="0" cellspacing="0">
    <TR>
        <TD><b>Name</b></TD>
        <TD><b>Size</b></TD>
        <TD><b>DateLastModified</b></TD>
    </TR>
    <CFOUTPUT QUERY="QueryDir">
        <TR>
            <TD>#Name#</TD>
            <TD>#Size#</TD>
            <TD>#DateLastModified#</TD>
        </TR>
    </CFOUTPUT>
</TABLE>
```

Здесь для вывода результатов списка каталогов и файлов был использован тег <CFOUTPUT> с атрибутом QUERY. В этом же примере были указаны атрибуты FILTER и SORT, позволяющие ограничивать и сортировать полученный список.

В качестве закрепления материала предлагаем читателю разработать собственную службу управления файловой системой, где бы можно было с легкостью обрабатывать файлы и каталоги, находящиеся на вашем компьютере, используя при этом возможности ColdFusion.

Резюме

Подведем итог. В данной главе мы использовали CFML-теги <CFFILE> и <CFDIRECTORY>.



Протоколы для работы с удаленными серверами

В *главе 17* мы уже познакомились с управлением файловой системой на локальном сервере. Однако возможности ColdFusion по работе с файлами на этом не заканчиваются. Как известно, для Internet/intranet существуют различные протоколы, позволяющие передавать и принимать с удаленных серверов необходимую информацию. ColdFusion в этом плане поддерживает следующие протоколы: HTTP (HyperText Transfer Protocol), FTP (File Transfer Protocol) и LDAP (Lightweight Directory Access Protocol). Одно из немаловажных преимуществ заключается в том, что перечисленные протоколы предоставляют пользователю возможность передавать файлы между двумя компьютерами на базе различных платформ.

Протокол передачи гипертекста

HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) является протоколом высокого уровня (а именно, уровня приложений), обеспечивающий необходимую скорость передачи данных, требующуюся для распределенных информационных систем. Ярким примером служит World Wide Web (WWW). HTTP-протокол может использоваться во многих задачах, таких как серверы имен и распределенные системы управления объектами, посредством расширения методов запроса. В ColdFusion применение HTTP-протокола для доступа к документам, расположенным на Web-сервере, происходит за счет использования тегов `<CFHTTP>` и `<CFHTTPPARAM>`.

Тег `<CFHTTP>`

```
<CFHTTP URL = "hostname"
        PORT = "port_number"
        METHOD = "Get | Post"
        USERNAME = "username"
        PASSWORD = "password"
        NAME = "queryname"
        COLUMNS = "query_columns">
```

```

PATH = "path"
FILE = "filename"
DELIMITER = "character"
TEXTQUALIFIER = "character"
RESOLVEURL = "Yes | No"
PROXYSERVER = "hostname"
PROXYPORT = "port_number"
USERAGENT = "user_agent"
THROWONERROR = "Yes | No"
REDIRECT = "Yes | No"
TIMEOUT = "timeout_period">

```

```
</CFHTTP>
```

В табл. 18.1 представлено описание атрибутов тега <CFHTTP>.

Таблица 18.1. Описание атрибутов тега <CFHTTP>

Атрибут	Описание
URL	Полный путь к ресурсу, включая название протокола HTTP или HTTPS. Обязательный атрибут
PORT	Номер порта. По умолчанию 80. Обязательный атрибут
METHOD	Метод работы. Возможные значения: • <code>Get</code> — позволяет получать текстовые файлы или разгружать (download) бинарные файлы; • <code>Post</code> — позволяет посылать информацию к странице сервера или CGI-программе для обработки. Данный метод предполагает использование тега <CFHTTPPARAM>. Обязательный атрибут
USERNAME	Имя пользователя. Атрибут, необходимый при аутентификации на сервере
PASSWORD	Пароль пользователя. Атрибут, необходимый при аутентификации на сервере
NAME	Имя запроса при получении файла. Необязательный атрибут
COLUMNS	Список колонок запроса, разделенных запятой. Применяется для извлечения данных из текстового файла, используя метод <code>Get</code> . При этом извлечение данных происходит, начиная со второй строки, а первая строка текстового файла воспринимается как заголовок. Указание данного атрибута подразумевает переопределение имен колонок запроса, по умолчанию принимающих значения заголовка текстового файла. Необязательный атрибут

Таблица 18.1 (продолжение)

Атрибут	Описание
PATH	Путь к каталогу, куда должен быть помещен файл, указанный с использованием атрибута FILE. Необязательный атрибут
FILE	Имя файла. Требуется при использовании метода Post, если указывается атрибут PATH. Впрочем, при указании PATH атрибут FILE также может быть использован с методом Get, если существует необходимость размещения файла с именем, отличным от того, к которому происходит обращение с помощью атрибута URL
DELIMITER	Разделитель, используемый для создания запроса. По умолчанию — запятая. Необязательный атрибут
TEXTQUALIFIER	Квалификатор, определяющий начало и конец колонки для формирования запроса. По умолчанию — двойные кавычки. Необязательный атрибут
RESOLVEURL	Индикатор поддержки ссылок. Положительное значение данного атрибута позволяет в переменную FileContent (содержимое файла) помимо самого содержимого документа включить номер порта, так чтобы связи остались неповрежденными. Возможные значения: • Yes — да; • No — нет (по умолчанию). Необязательный атрибут
PROXYSERVER	Наименование хоста или IP-адрес проху-сервера. Необязательный атрибут
PROXYPORT	Номер порта проху-сервера. По умолчанию — 80. Необязательный атрибут
USERAGENT	Пользовательский агент запрашиваемых заголовков. Обязательный атрибут
THROWONERROR	Индикатор контроля ошибок. Указывает на возможность вызова исключительной ситуации, которая может быть обработана с помощью предназначенных для этой цели тегов <CFTRY> и <CFCATCH>. Допустимые значения: • Yes — да; • No — нет (по умолчанию). Необязательный атрибут
REDIRECT	Индикатор переадресации. Указывает, совершать ли переадресацию выполнения или сделать остановку в непредвиденной ситуации. ColdFusion поддерживает до пяти переадресаций.

Таблица 18.1 (окончание)

Атрибут	Описание
TIMEOUT	Возможные значения:
	• Yes — да (по умолчанию); • No — нет.
	Необязательный атрибут
TIMEOUT	Период ожидания для соединения с Web-сервером. Определяется в секундах. Необязательный атрибут

При использовании тега <CFHTTP> создается объект cfhttp, обеспечивающий доступ к вспомогательной информации, посредством следующих переменных:

- ☐ cfhttp.FileContent — содержимое файла для текстовых документов и MIME-файлов;
- ☐ cfhttp.MimeType — тип MIME-файла;
- ☐ cfhttp.StatusCode — состояние выполнения HTML-страницы, при индикаторе контроля ошибок (атрибут THROWONERROR), находящемся в позиции отключения вызова исключительной ситуации, содержит код ошибки с соответствующим примечанием;
- ☐ cfhttp.Header — строка заголовка возвращаемого документа;
- ☐ cfhttp.ResponseHeader[http_hd_key] — заголовок возвращаемого документа в виде массива в пределах структуры (structure).

Тег <CFHTTPPARAM>

Синтаксис:

```
<CFHTTPPARAM NAME = "name"
               TYPE = "type"
               VALUE = "transaction type"
               FILE = "filename">
```

В табл. 18.2 представлено описание атрибутов тега <CFHTTPPARAM>.

Таблица 18.2. Описание атрибутов тега <CFHTTPPARAM>

Атрибут	Описание
NAME	Имя передаваемой переменной. Обязательный атрибут

Таблица 18.2 (окончание)

Атрибут	Описание
TYPE	Тип передаваемой переменной. Возможные значения: • URL — переменная URL; • FormField — поле формы; • Cookie — переменная cookies; • CGI — переменная окружения CGI; • File — файл. Обязательный атрибут
VALUE	Значение передаваемой переменной. Необязательный атрибут для типа File
FILE	Полное имя файла, включая путь. Обязательный атрибут для типа File

Примеры работы с удаленным сервером через HTTP-протокол

Предположим, что нам необходимо получить содержимое страницы login.cfm, находящейся на Web-сервере ipweb в подкаталоге mail. Для решения данной задачи мы воспользуемся тегом <CFHTTP> и методом Get:

```
<CFHTTP METHOD = "Get"
    URL = "http://ipweb/mail/login.cfm"
    RESOLVEURL = "Yes">
<CFOUTPUT>#cfhttp.FileContent#</CFOUTPUT>
```

Здесь указан атрибут RESOLVEURL, который, как уже было отмечено в описании тега <CFHTTP>, позволяет в переменную FileContent (содержимое файла) помимо самого содержимого документа включить номер порта, так чтобы связи остались неповрежденными. Затем на экран выводится содержимое полученной страницы с использованием все той же внутренней переменной FileContent.

Еще один пример с применением метода Get позволяет сохранять полученный документ на сервере ColdFusion:

```
<CFHTTP METHOD = "Get"
    URL = "http://ipweb/mail/login.cfm"
    PATH = "c:\Apache\htdocs\my-app\"
    FILE = "login.html"
```

```
USERNAME = "IUSR_IPWEB"
```

```
PASSWORD = "*****">
```

```
<CFOUTPUT>#cfhttp.StatusCode#</CFOUTPUT>
```

где атрибут `PATH` устанавливает путь к каталогу сервера ColdFusion, в который должен быть помещен документ с именем, указанным с помощью атрибута `FILE`. При этом используются необязательные атрибуты `USERNAME` и `PASSWORD`, позволяющие идентифицировать соединение с удаленным сервером. Здесь же мы выводим на экран состояние выполнения тега `<CFHTTP>` с помощью переменной `StatusCode`, что позволяет удостовериться в успешном выполнении кода при значении этой переменной, равном `"200 Success"`.

Тег `<CFHTTP>` также позволяет создавать запросы, используя текстовые файлы, отредактированные по определенному правилу:

- ❑ начало и конец каждой колонки данных должны быть отделены специальными символами (атрибут `TEXTQUALIFIER`, по умолчанию — двойные кавычки);
- ❑ между колонками должен присутствовать символ разделения (атрибут `DELIMITER`, по умолчанию — запятая);
- ❑ первая строка данных воспринимается как заголовок;
- ❑ каждая последующая строка рассматривается как очередная запись данных.

В качестве примера создадим файл `TestDataSources.txt` (листинг 18.1), который будет являться источником для запроса. Обращаем ваше внимание, что данные перечисляются через точку с запятой (;), о чем в последствии надо не забыть.

Листинг 18.1. Текстовый файл `TestDataSources.txt`

```
"A";"B";"C"
```

```
"Mark";"Portugal";"1962-10-30"
```

```
"Mike";"Stanley";"1963-06-25"
```

```
"Noman";"GarciaParra";"1973-07-23"
```

```
"Pat";"Rapp";"1967-07-13"
```

```
"Pedro";"Martinez";"1971-10-25"
```

Теперь, когда источник данных существует в форме текстового файла, сформируем код страницы, позволяющей создавать запрос на основе имеющихся данных и отражать эти данные в окне обозревателя в виде таблицы, как показано в листинге 18.2.

Листинг 18.2. Код страницы MyCFHttp.cfm

```

<HTML>
<HEAD>
  <TITLE>CFHTTP</TITLE>
</HEAD>
<BODY>
  <CFHTTP METHOD = "Get"
    URL = "http://ipweb/TestDateSources.txt"
    NAME = "MyCFHttp"
    COLUMNS = "FName,LName,BDay"
    DELIMITER = ";">
<TABLE cellpadding="0" cellspacing="8">
  <TR>
    <TD><b>FirstName</b></TD>
    <TD><b>LastName</b></TD>
    <TD><b>BirthDay</b></TD>
  </TR>
  <TR><TD bgcolor="123456" colspan="3" height=1></TD></TR>
  <CFOUTPUT QUERY="MyCFHttp">
    <TR>
      <TD>#FName#</TD><TD>#LName#</TD><TD>#BDay#</TD>
    </TR>
  </CFOUTPUT>
</TABLE>
</BODY>
</HTML>

```

Если бы в текстовом файле данные были записаны через запятую, то атрибут `DELIMITER` можно было бы не использовать. Атрибут `COLUMNS` также можно не указывать, но в таком случае в качестве имен переменных необходимо учитывать значения колонок из первой строки текстового файла, так называемый заголовок данных. Для нашего примера это `A`, `B` и `C`. На рис. 18.1 показан результат выполнения кода приведенного выше примера.

Все предыдущие примеры использования тега `<CFHTTP>` основывались на методе `Get`. Теперь приведем пример с применением метода `Post`, позволяющего отправлять данные странице сервера или CGI-программе для обработки. Такой метод предполагает использование тега `<CFHTTPPARAM>`, синтаксис которого достаточно прост, в чем уже можно было убедиться в данной главе.

FirstName	LastName	Birthday
Mark	Portugal	1962-10-30
Mike	Stanley	1963-06-25
Noman	Garciaparra	1973-07-23
Pat	Rapp	1967-07-13
Pedro	Martinez	1971-10-25

Рис. 18.1. Результат выполнения кода MyCFHttp.cfm

Предположим, нам необходимо отправить два параметра для выполнения одной из страниц удаленного сервера. Пусть один из параметров является переменной URL, а в качестве второго параметра передается бинарный файл. В свою очередь первый параметр выступает в качестве содержания создаваемого текстового файла, а второй параметр предназначен для загрузки файла на удаленный сервер. При этом надо отдавать себе отчет, что на удаленной машине также должен быть установлен и запущен ColdFusion Server. Приведем пример кода, позволяющего решить поставленную задачу (листинг 18.3).

Листинг 18.3. Код страницы MyCFHttpPost.cfm

```
<HTML>
<HEAD>
  <TITLE>CFHTTP</TITLE>
</HEAD>
<BODY>
  <CFHTTP METHOD = "Post"
    URL = "http://ipweb/cfipweb/test.cfm">
    <CFHTTPPARAM TYPE = "URL"
      NAME = "City"
      VALUE = "NewYork">
    <CFHTTPPARAM TYPE = "File"
      NAME = "filePhoto"
      FILE = "D:\MyPhoto\Harlem.gif">
  </CFHTTP>
</BODY>
</HTML>
```


Так как в качестве второго параметра мы отправляем файл, что следует из значения атрибута `TYPE` тега `<CFHTTPPARAM>`, то здесь же мы указываем расположение этого файла с помощью атрибута `FILE` того же тега `<CFHTTPPARAM>`. В листинге 18.4 приведен пример страницы `Test.cfm`, в которую и передаются указанные параметры.

Листинг 18.4. Код страницы `Test.cfm`

```
<HTML>
<HEAD>
  <TITLE>Test</TITLE>
</HEAD>
<BODY>
  <CFIF isDefined("URL.City")>
    <CFFILE ACTION="Write"
      FILE="C:\InetPub\wwwroot\city.txt"
      OUTPUT="#URL.City#">
  </CFIF>
  <CFIF isDefined("Form.filePhoto")>
    <CFFILE ACTION="Upload"
      FILEFIELD="Form.filePhoto"
      DESTINATION="C:\InetPub\wwwroot\"
      NAMECONFLICT="Overwrite">
  </CFIF>
</BODY>
</HTML>
```

Совершаемые на этой странице действия подробно описаны в *главе 17*. Отметим только, что данный файл должен находиться на удаленном сервере `ipweb` (каталог `C:\InetPub\wwwroot` есть ни что иное, как приемник бинарного файла, находящийся на удаленном сервере `ipweb`), и доступ к переменной, определяющей переданный бинарный файл, осуществляется через приставку `Form`.

Протокол передачи файлов

Протокол FTP (File Transfer Protocol, протокол передачи файлов) является одним из ранних протоколов в Internet и входит в его стандарты. Обмен данными по протоколу FTP проходит по ТСП-каналу. Построен обмен по технологии «клиент-сервер». FTP-протокол дает возможность пользователям

одного компьютера получать доступ к файловой системе другого компьютера и получать/передавать файлы между компьютерами.

Существует два способа организации доступа к файловой системе удаленного компьютера по FTP-протоколу: *анонимный* и *авторизованный*.

Анонимный доступ обеспечивается под видом специально выделенного пользователя, которого еще называют `anonymous`, с необязательным паролем, впрочем, правило "хорошего тона" — в качестве пароля использовать адрес электронной почты. При входе в систему с правами `anonymous` вы получаете доступ к специально выделенному для этих пользователей каталогу, который называется FTP-сервером.

Что касается авторизованного доступа, то на конкретной удаленной системе осуществлять подобный доступ имеют право только пользователи этой удаленной системы после прохождения авторизации. После входа в систему пользователь, как правило, получает доступ к своему домашнему каталогу и ко всем другим файловым ресурсам удаленной системы, к которым он имеет права на доступ. Доступ же к FTP-серверу в ColdFusion осуществляется с помощью тега `<CFFTP>`.

Прежде чем приводить синтаксис и описание тега `<CFFTP>`, необходимо оговорить некоторые нюансы. Тег `<CFFTP>` предполагается использовать в двух режимах, первый из которых позволяет открывать и закрывать доступ к файлам и каталогам на FTP-сервере, а второй предназначен для непосредственной работы с файловой системой на FTP-сервере в период установленного соединения. После инициализации соединения открытый канал может быть использован как для приема, так и для передачи данных.

Канал управления должен быть открыт при передаче данных между машинами. В случае его закрытия передача данных прекращается. Но не забывайте закрывать канал управления принудительно после совершения необходимых действий, т. к. FTP-протокол серьезно загружает каналы, соединяющие вас с другими сетями.

Тег `<CFFTP>` для открытия и закрытия соединения

Синтаксис:

```
<CFFTP ACTION = "Open | Close"
    USERNAME = "name"
    PASSWORD = "password"
    SERVER = "server"
    TIMEOUT = "timeout in seconds"
    PORT = "port"
```

```

CONNECTION = "name"
PROXYSERVER = "proxyserver"
RETRYCOUNT = "number"
STOPONERROR = "Yes | No"
PASSIVE = "Yes | No">

```

В табл. 18.3 приведено описание атрибутов тега <CFFTP> (вариант 1).

Таблица 18.3. Описание атрибутов тега <CFFTP> (вариант 1)

Атрибут	Описание
ACTION	Действие. Возможные значения: • Open — открытие сессии; • Close — закрытие сессии. Обязательный атрибут
USERNAME	Имя пользователя. Обязательный атрибут для действия Open
PASSWORD	Пароль пользователя. Обязательный атрибут для действия Open
SERVER	Наименование FTP-сервера. Обязательный атрибут для действия Open
TIMEOUT	Ожидаемый период времени для соединения с FTP-сервером. Определяется в секундах. По умолчанию равен 30 секундам. Необязательный атрибут
PORT	Номер порта FTP-сервера. По умолчанию равен 21. Необязательный атрибут
CONNECTION	Название соединения. При повторном применении тега <CFFTP> использует существующую связь. Необязательный атрибут
PROXYSERVER	Наименование прокси-сервера. Необязательный атрибут
RETRYCOUNT	Число повторных попыток до возникновения сообщения об ошибке. По умолчанию равен 1. Необязательный атрибут
STOPONERROR	Индикатор прерывания при возникновении ошибок. Возможные значения: • Yes — да (по умолчанию); • No — нет. Необязательный атрибут
PASSIVE	Индикатор пассивной работы. Возможные значения: • Yes — да; • No — нет (по умолчанию). Необязательный атрибут

При использовании тега `<CFFTP>` с атрибутом `STOPONERROR`, равным `No`, создается объект `cffftp`, предоставляющий вспомогательную информацию относительно работы FTP-сервера с помощью следующих переменных:

- ☐ `cffftp.succeeded` — индикатор успешного доступа, принимает значения `Yes` или `No`;
- ☐ `cffftp.errorCode` — номер ошибки, если она возникает;
- ☐ `cffftp.errorText` — сообщение об ошибке.

Тег `<CFFTP>` для управления файловой системы FTP-сервера

Синтаксис:

```
<CFFTP ACTION = "action"
    USERNAME = "name"
    PASSWORD = "password"
    NAME = "query_name"
    SERVER = "server"
    TIMEOUT = "timeout in seconds"
    PORT = "port"
    CONNECTION = "name"
    ASCIIEXTENSIONLIST = "extensions"
    TRANSFERMODE = "ASCII | Binary | Auto"
    FAILIFEXISTS = "Yes | No"
    DIRECTORY = "directory name"
    LOCALFILE = "filename"
    REMOTEFILE = "filename"
    ITEM = "directory or file"
    EXISTING = "file or directory name"
    NEW = "file or directory name"
    PROXYSERVER = "proxyserver"
    RETRYCOUNT = "number"
    STOPONERROR = "Yes | No"
    PASSIVE = "Yes | No">
```

В табл. 18.4 представлено описание атрибутов тега `<CFFTP>` (вариант 2).

Таблица 18.4. Описание атрибутов тега `<CFFTP>` (вариант 2)

Атрибут	Описание
ACTION	Действие, позволяющее управлять файловой системой. Возможные значения: • <code>ListDir</code> — создает объект запроса со списком каталогов и файлов текущего каталога; • <code>ChangeDir</code> — изменяет текущий каталог; • <code>CreateDir</code> — создает каталог; • <code>RemoveDir</code> — удаляет каталог; • <code>GetFile</code> — получает файл; • <code>PutFile</code> — помещает файл; • <code>Rename</code> — переименовывает файл или каталог; • <code>Remove</code> — удаляет файл; • <code>GetCurrentDir</code> — получает имя текущего каталога; • <code>GetCurrentURL</code> — закрывает сессию; • <code>ExistsDir</code> — выясняет существование каталога; • <code>ExistsFile</code> — выясняет существование файла; • <code>Exists</code> — выясняет существование объекта. Обязательный атрибут
NAME	Имя запроса. Обязательный атрибут для действия <code>ListDir</code>
ASCIIEXTENSIONLIST	Список расширений файлов, которые передаются как ASCII-коды. По умолчанию принимает значение <code>txt;htm;html;cfm;cfml;shtm;shtml;css;asp;asa</code> при условии, что указан атрибут <code>TRANSFERMODE='Auto'</code>. Необязательный атрибут
TRANSFERMODE	Способ передачи файлов. Обязательный атрибут для действий <code>GetFile</code> и <code>PutFile</code>. Возможные значения: • <code>ASCII</code> — ASCII-коды; • <code>Binary</code> — бинарный; • <code>Auto</code> — определяется автоматически
FAILIFEXISTS	Определяет, будет ли игнорироваться действие <code>GetFile</code>, если такой файл уже существует. Возможные значения: • <code>Yes</code> — да (по умолчанию); • <code>No</code> — нет

Таблица 18.4 (окончание)

Атрибут	Описание
DIRECTORY	Каталог, по отношению к которому производится действие. Обязательный атрибут для действий <code>ChangeDir</code> , <code>CreateDir</code> , <code>ListDir</code> и <code>ExistsDir</code>
LOCALFILE	Имя файла на локальной машине. Обязательный атрибут для действий <code>GetFile</code> и <code>PutFile</code>
REMOTEFILE	Имя файла на удаленном FTP-сервере. Обязательный атрибут для действий <code>GetFile</code> , <code>PutFile</code> , <code>ExistsFile</code>
ITEM	Определяет объект, файл или каталог для указанного действия. Обязательный атрибут для действий <code>Exists</code> и <code>Remove</code>
EXISTING	Текущее имя файла или каталога на удаленном сервере. Обязательный атрибут для действия <code>Rename</code>
NEW	Новое имя файла или каталога на удаленном сервере. Обязательный атрибут для действия <code>Rename</code>

При использовании тега `<CFFTP>` с действиями `GetCurrentDir`, `GetCurrentURL`, `ExistsDir`, `ExistsFile` и `Exists` ColdFusion создает переменную `cffftp.returnValue`, значение которой зависит от используемого действия. Приведем список возможных значений переменной `cffftp.returnValue` в зависимости от применяемого действия:

- ☐ `GetCurrentDir` — имя текущего каталога;
- ☐ `GetCurrentURL` — текущий адрес URL;
- ☐ `ExistsDir` — Yes или No в зависимости от результата поиска каталога;
- ☐ `ExistsFile` — Yes или No в зависимости от результата поиска файла;
- ☐ `Exists` — Yes или No в зависимости от результата поиска объекта.

Следует также отметить, что при действии `ListDir` тег `<CFFTP>` создает объект запроса, обеспечивающий доступ к вспомогательной информации через следующие переменные:

- ☐ `queryName.Name` — имя каталога или файла текущего элемента;
- ☐ `queryName.Path` — путь к файлу или каталогу без указания полного пути;
- ☐ `queryName.URL` — адрес URL для текущего файла или каталога;
- ☐ `queryName.Length` — размер файла текущего элемента;

- ❑ `queryName.LastModified` — дата и время последнего изменения текущего элемента;
- ❑ `queryName.Attributes` — символьная строка, содержащая значение `Normal` или `Directory` в зависимости от типа элемента (файл или каталог);
- ❑ `queryName.IsDirectory` — индикатор, указывающий, является ли элемент каталогом, и принимающий значения `Yes` или `No`;
- ❑ `queryName.Mode` — доступ, в Windows не используется.

Пример реализации упрощенного FTP-клиента

В качестве примера рассмотрим создание собственного FTP-клиента, с одной стороны имеющего достаточно ограниченные возможности, но при этом позволяющего продемонстрировать принцип работы с уже описанным тегом `<CFFTP>`.

Любой FTP-сервер, прежде чем совершить какие-либо действия по отношению к файлам или каталогам, предполагает предварительное открытие доступа, где в качестве атрибутов регистрации необходимо использовать `SERVER`, `USERNAME` и `PASSWORD`. Поскольку наш FTP-клиент должен иметь возможность регистрации на разных FTP-серверах, необходимо использовать какой-либо источник со списком возможных соединений. Во всех предыдущих примерах для подобных целей мы использовали базу данных, здесь же для упрощения реализации данной задачи возьмем обычный текстовый файл с именем `RegisrationFTP.txt`. Причем помимо имени сервера, имени пользователя и пароля в данном файле регистрации будем хранить имя каталога на локальном сервере, который должен являться источником файлов для копирования на FTP-сервер.

Как мы уже отмечали, FTP-протокол серьезно загружает каналы, соединяющие вас с другими сетями, по этой причине в нашей программе любое действие с файлами на FTP-сервере будет предваряться открытием соединения и закрытием сразу по завершении указанного действия.

На рис. 18.2 показана схема взаимодействия модулей системы FTP-клиента.

На приведенной схеме штрих-пунктирной линией обозначен модуль, не рассматриваемый в описываемых ниже кодах.

В данной главе уже приводился пример считывания данных из текстового файла, точно таким же образом мы считываем необходимые данные из файла `RegistrationFTP.txt`, где изначально выбираются сведения из первой записи. Напомним, что первой записью является перечень полей для второй и следующих строк текстового файла (листинг 18.5).

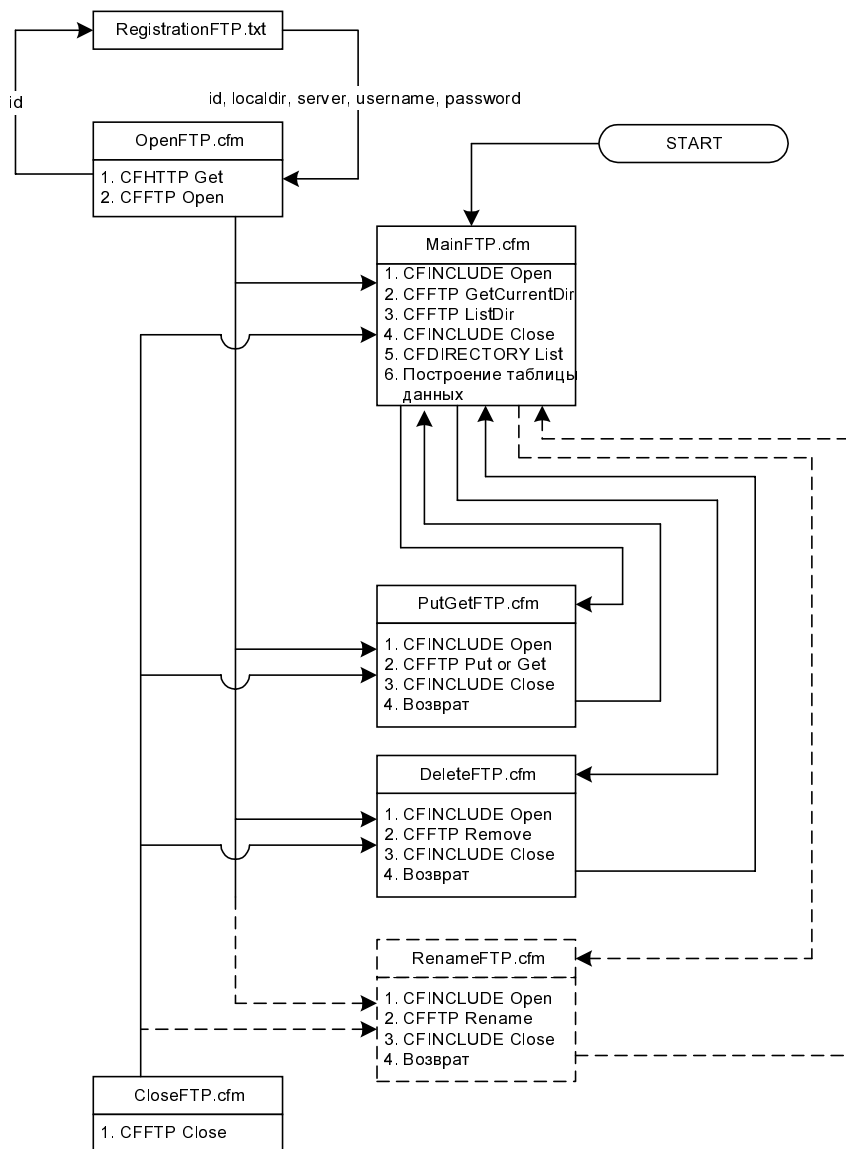


Рис. 18.2. Схема взаимодействия модулей системы FTP-клиента

Листинг 18.5. Текстовый файл RegistrationFTP.txt

```

"ID","LocalDir","Server","UserName","Password"
"1","C:\SourceFTP\IPWeb\","IPWeb","IUSR_IPWEB","*****"

```



```
"2","C:\SourceFTP\Allaire\","ftp.allaire.com","Anonymous","Ruben@MyMail"  
"3","C:\SourceFTP\Allaire\","ftp2.allaire.com","Anonymous","Ruben@MyMail"
```

Примечание

Звездочками (*) автор скрывает собственный пароль.

В качестве следующего кода (листинг 18.6) приводится пример считывания данных для регистрации и открытия сеанса работы с FTP-сервером.

Листинг 18.6. Получение данных для регистрации и соединения, файл OpenFTP.cfm

```
<CFHTTP METHOD = "Get"  
    URL = "http://127.0.0.1/myftp/registrationftp.txt"  
    NAME = "RegFTP">  
  
<CFOUTPUT QUERY = "RegFTP"  
    MAXROWS = "1" STARTROW = "#CurIDConnect#">  
  
    <CFFTP ACTION = "Open"  
        CONNECTION = "FTP"  
        SERVER = "#Server#"  
        USERNAME = "#UserName#"  
        PASSWORD = "#Password#"  
        RETRYCOUNT = "3">  
  
    <CFSET LocalDir = LocalDir>  
</CFOUTPUT>
```

Здесь переменная `CurIDConnect` заранее определяется на загружаемой странице. В частности, одной из таких страниц, как это видно из схемы взаимодействия модулей, является страница `MainFTP.cfm`. Она же и рассматривается в качестве главной страницы всего проекта.

Что же происходит при выполнении кода `MainFTP.cfm`. В первую очередь анализируется наличие необходимой переменной для поиска данных, позволяющих произвести регистрацию на FTP-сервере. В случае отсутствия этой переменной ей присваивается значение по умолчанию, равное 1. Далее происходит регистрация и непосредственное установление соединения с FTP-сервером. Затем определяется текущий каталог на FTP-сервере с помощью тега `<CFFTP>` с действием `GetCurrentDir` и с указанием имени открытого соединения. Следующим шагом, используя все тот же тег `<CFFTP>` и то же соединение, но уже с действием `ListDir`, создается запрос `ListLocalDir`, в который помещается список файлов из текущего каталога

FTP-сервера. Надо заметить, что при выводе на экран мы ограничиваем данный список, показывая только файлы с расширением cfm без отображения подкаталогов. На рис. 18.3 изображено рабочее состояние программы FTP-клиента, где описываемый список файлов расположен в виде таблицы справа. В левой же таблице находится список файлов, полученных в результате использования тега `<CFDIRECTORY>` с действием `List` и указанием текущего каталога с помощью переменной `LocalDir`, определяемой из регистрационного файла `RegistrationFTP.txt`. Но перед формированием таблиц с данными списками на странице `MainFTP.cfm` производится плановое закрытие соединения в результате использования тега `<CFFTP>` с действием `Close`. Листинг страницы `MainFTP.cfm` представлен ниже.

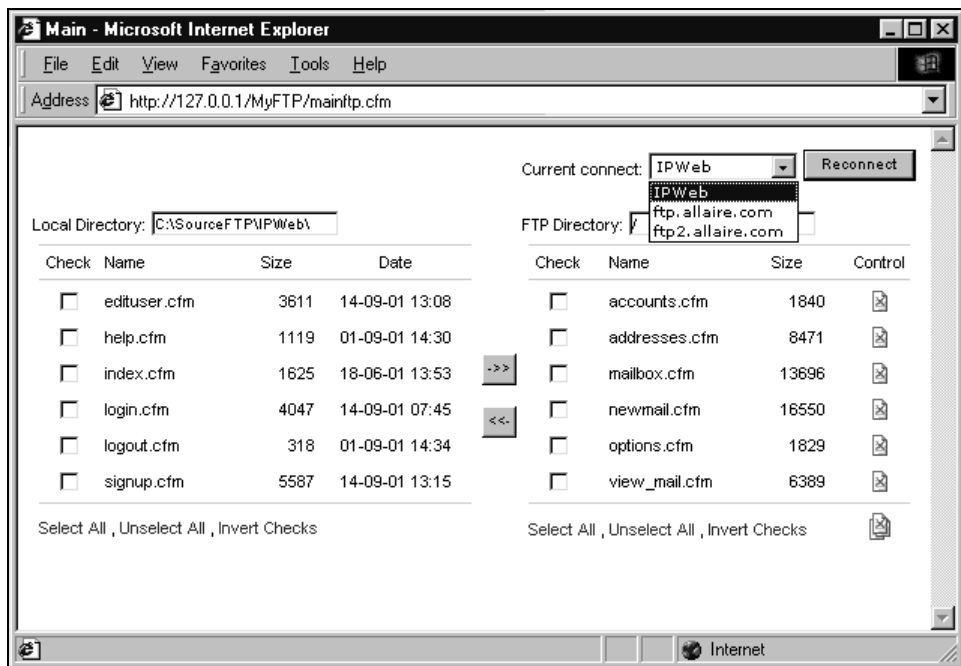


Рис. 18.3. Пример работы FTP-клиента, страница `MainFTP.cfm`

Листинг 18.7. Код страницы `MainFTP.cfm`

```
<HTML>
<HEAD>
  <TITLE>Main</TITLE>
</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
```

```
<script type="text/javascript" language="JavaScript"
    src="selectchecks.js"></script>
<script>
function PutGetDelSelFile(form, type, curidconnect) {
    idxlen = SelUnselInv(form);
    var strCheck = "";
    for (i=0;i<idxlen;i++) {
        if(ns) {
            eval("var valItem = document."+form+".elements[i].checked");
            if (valItem==true) {
                if (strCheck=="") {
                    eval("strCheck = document."+form+".elements[i].value");
                } else {
                    eval("strCheck = strCheck + ',' +
                        document."+form+".elements[i].value");
                }
            }
        }
        } else {
            eval("var valItem = document."+form+".elements(i).checked");
            if (valItem==true) {
                if (strCheck=="") {
                    eval("strCheck = document."+form+".elements(i).value");
                } else {
                    eval("strCheck = strCheck + ',' +
                        document."+form+".elements(i).value");
                }
            }
        }
    }
    if (strCheck == "") {
        alert("For put or get it is necessary to select even one file!");
    } else {
        if (type == "Del") {
            eval("window.location.href =
                'deleteftp.cfm?Type="+type+"&CurIDConnect="+
                curidconnect+"&NameSel="+strCheck+"'");
        } else {
```

```

eval("window.location.href =
    'putgetftp.cfm?Type="+type+"&CurIDConnect="+
    curidconnect+"&NameSel="+strCheck+"'");
    }
}
}
</script>
<CFIF isDefined("IDConnect")>
    <CFSET CurIDConnect = IDConnect>
<CFELSE>
    <CFSET CurIDConnect = 1>
</CFIF>
<CFINCLUDE TEMPLATE = "openftp.cfm">
<CFFTP ACTION = "GetCurrentDir"
    CONNECTION = FTP>
<CFSET CurDirFTP = cffftp.ReturnValue>
<CFFTP ACTION = "ListDir"
    CONNECTION = "FTP"
    DIRECTORY = "#CurDirFTP#"
    NAME="ListCurDirFTP">
<CFINCLUDE TEMPLATE = "closeftp.cfm">
<CFDIRECTORY ACTION = "List"
    DIRECTORY = "#LocalDir#"
    NAME = "ListLocalDir"
    FILTER = "*.cfm"
    SORT = "name ASC">
<TABLE width="100%" cellpadding="0" cellspacing="0">
    <TR><TD colspan="3" class="txt" align="right">
        <CFFORM NAME = "Form" ACTION = "mainftp.cfm">
            Current connect:
            <CFSELECT QUERY = "RegFTP" NAME = "IDConnect"
                VALUE = "ID" DISPLAY = "Server"
                SELECTED = "#CurIDConnect#">
            </CFSELECT>
            <input type="Submit" value="Reconnect">
        </CFFORM>
    </TD></TR>
<TR>

```

```

<TD class="txt">Local Directory:
    <input name="LocalDir" size="28" readonly="Yes"
        value="<CFOUTPUT>#LocalDir#</CFOUTPUT>"></TD>
<TD></TD>
<TD class="txt">FTP Directory:
    <input name="CurDirFTP" size="28" readonly="Yes"
        value="<CFOUTPUT>#CurDirFTP#</CFOUTPUT>"></TD>
</TR>
<TR>
    <TD valign="top">
        <TABLE width="100%" cellpadding="2" cellspacing="4">
            <TR><TD colspan="4" bgcolor="#cbccbd" height=1</TD></TR>
            <TR>
                <TD class="txt" width="40" align="center">Check</TD>
                <TD class="txt">Name</TD>
                <TD class="txt" width="50" align="center">Size</TD>
                <TD class="txt" width="100" align="center">Date</TD>
            </TR>
            <TR><TD colspan="4" bgcolor="#cbccbd" height=1</TD></TR>
            <FORM method="post" name="form1">
                <CFOUTPUT QUERY="ListLocalDir">
                    <TR>
                        <TD class="txt" width="40" align="center">
                            <input type="checkbox" value="#Name#" name="#Name#">
                        </TD>
                        <TD class="txt">#Name#</TD>
                        <TD class="txt" width="50" align="right">#Size#</TD>
                        <TD class="txt" width="100"
                            align="center">#DateFormat(DateLastModified,
                                "dd-mm-yy")# #TimeFormat(DateLastModified,
                                    "HH:mm")#</TD>
                    </TR>
                </CFOUTPUT>
            </FORM>
            <TR><TD colspan="4" bgcolor="#cbccbd" height=1</TD></TR>
            <TR>
                <TD class="txt" colspan="4" height="25">
                    <a href="javascript:SelectAll('form1');"
                        class="txtlink">Select All</a>&nbsp;  ;

```

```

        <a href="javascript:UnselectAll('form1');"
            class="txtlink">Unselect All</a>&nbsp;  ;
        <a href="javascript:InvertChecks('form1');"
            class="txtlink">Invert Checks</a>
    </TD>
</TR>
</TABLE>
</TD>
<TD width="30" align="center" class="txt">
    <input type="Button" value="->"
        onClick="javascript:PutGetDelSelFile('form1', 'Put',
            '<CFOUTPUT>#CurIDConnect#</CFOUTPUT>');">
    <br><br>
    <input type="Button" value="<->"
        onClick="javascript:PutGetDelSelFile('form2', 'Get',
            '<CFOUTPUT>#CurIDConnect#</CFOUTPUT>');">
</TD>
<TD valign="top">
    <TABLE width="100%" cellpadding="2" cellspacing="4">
        <TR><TD colspan="4" bgcolor="#cbccbd" height=1></TD></TR>
        <TR>
            <TD class="txt" width="40" align="center">Check</TD>
            <TD class="txt">Name</TD>
            <TD class="txt" width="50" align="center">Size</TD>
            <TD class="txt" width="40" align="center">Control</TD>
        </TR>
        <TR><TD colspan="4" bgcolor="#cbccbd" height=1></TD></TR>
        <FORM method="post" name="form2">
            <CFOUTPUT QUERY="ListCurDirFTP">
            <CFIF IsDirectory eq "No" AND Right(Name, 4) eq ".cfm">
            <TR>
                <TD class="txt" width="40" align="center">
                    <input type="checkbox" name="#Name#" value="#Name#">
                </TD>
                <TD class="txt">#Name#</TD>
                <TD class="txt" width="50" align="right">#Length#</TD>
                <TD class="txt" width="40" align="center">

```

```

        <a
href="deleteftp.cfm?Name=#Name#&CurIDConnect=#CurIDConnect#">
        <IMG src="delete.gif" alt="Delete" border='0'></a></TD>
    </TD>
</TR>
</CFIF>
</CFOUTPUT>
</FORM>
<TR><TD colspan="4" bgcolor="#cbccbd" height=1></TD></TR>
<TR>
    <TD class="txt" colspan="3" height="25">
        <a href="javascript:SelectAll('form2');"
            class="txtlink">Select All</a>&nbsp;&nbsp;&nbsp;;
        <a href="javascript:UnselectAll('form2');"
            class="txtlink">Unselect All</a>&nbsp;&nbsp;&nbsp;;
        <a href="javascript:InvertChecks('form2');"
            class="txtlink">Invert Checks</a>
    </TD>
    <TD align="center">
        <a href="javascript:PutGetDelSelFile('form2', 'Del',
            '<CFOUTPUT>#CurIDConnect#</CFOUTPUT>');"
            <IMG src="deletesel.gif" border='0'
                alt="Delete Selected"></a>
    </TD>
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

Как вы, наверное, уже заметили, в данную страницу включен файл JavaScript `SelectChecks.js` (листинг 18.8), содержащий дополнительные функции, необходимые для правильного выделения флажков (checks) с помощью инструментов группового выделения. Инструменты группового выделения показаны на рис. 18.3 (**Select All** (Выделить все), **Unselect All** (Снять выделение), **Invert Checks** (Инвертировать пометку)).

Листинг 18.8. Код JavaScript SelectChecks.js

```
function SelUnselInv(form) {
    eval("var idxlen = document."+form+".elements.length");
    return idxlen;
}

function SelectAll(form) {
    idxlen = SelUnselInv(form);
    for (i=0;i<idxlen;i++) {
        if(ns) eval("document."+form+".elements[i].checked = true");
        else    eval("document."+form+".elements(i).checked = true");
    }
}

function UnselectAll(form) {
    idxlen = SelUnselInv(form);
    for (i=0;i<idxlen;i++) {
        if(ns) eval("document."+form+".elements[i].checked = false");
        else    eval("document."+form+".elements(i).checked = false");
    }
}

function InvertChecks(form) {
    idxlen = SelUnselInv(form);
    for (i=0;i<idxlen;i++) {
        if(ns) {
            eval("var valItem = document."+form+".elements[i].checked");
            if (valItem==true) {
                eval("document."+form+".elements[i].checked = false");
            } else {
                eval("document."+form+".elements[i].checked = true");
            }
        } else {
            eval("var valItem = document."+form+".elements(i).checked");
            if (valItem==true) {
                eval("document."+form+".elements(i).checked = false");
            } else {
                eval("document."+form+".elements(i).checked = true");
            }
        }
    }
}
```


Во все страницы, так или иначе использующие FTP-соединение, включена страница CloseFTP.cfm, где приведен код для закрытия текущего соединения с FTP-сервером. Листинг страницы CloseFTP.cfm представлен ниже.

Листинг 18.9. Закрытие соединения. Файл CloseFTP.cfm

```
<CFFTP ACTION = "Close"
      CONNECTION = "FTP">
```

Следующая описываемая нами страница PutGetFTP.cfm (листинг 18.10) позволяет отправлять (put) и получать (get) файлы в зависимости от значения переменной Type, которое определяется одной из ссылок на главной странице приложения MainFTP.cfm.

Листинг 18.10. Отправление и получение файлов. Файл PutGetFTP.cfm

```
<HTML>
<HEAD>
  <TITLE>PutGet</TITLE>
</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGCOLOR#</cfoutput>">
  <CFIF isDefined("URL.Type") AND isDefined("URL.CurIDConnect")
    AND isDefined("URL.NameSel")>
    <CFINCLUDE TEMPLATE = "openftp.cfm">
    <CFSWITCH EXPRESSION=#Type#>
      <CFCASE VALUE="Put">
        <CFLOOP INDEX = "File" LIST="#URL.NameSel#" >
          <CFSET FullPath = LocalDir & File>
          <CFFTP ACTION = "PutFile"
                CONNECTION = "FTP"
                LOCALFILE = "#FullPath#"
                REMOTEFILE = "#File#"
                TRANSFERMODE = "Auto">
        </CFLOOP>
      </CFCASE>
      <CFCASE VALUE="Get">
        <CFLOOP INDEX = "File" LIST="#URL.NameSel#" >
          <CFSET FullPath = LocalDir & File>
          <CFFTP ACTION = "GetFile"
                CONNECTION = "FTP"
```

```

        LOCALFILE = "#FullPath#"
        REMOTEFILE = "#File#"
        TRANSFERMODE = "Auto">

    </CFLOOP>
</CFCASE>
</CFSWITCH>
<CFINCLUDE TEMPLATE = "closeftp.cfm">
</CFIF>
<META HTTP-EQUIV="Refresh"
    CONTENT="0;URL=mainftp.cfm?CurIDConnect=<CFOUTPUT>#CurIDConnect#
    </CFOUTPUT>">
</BODY>
</HTML>

```

И последней описываемой в этой главе является страница DeleteFTP.cfm (листинг 18.11). Уже судя по названию, можно догадаться о ее предназначении. Именно эта страница позволяет нам удалять файлы, находящиеся на FTP-сервере, причем как по отдельности, так и списком, заранее определенным все на той же главной странице приложения MainFTP.cfm.

Листинг 18.11. Удаление файлов на FTP-сервере. Файл DeleteFTP.cfm

```

<HTML>
<HEAD>
    <TITLE>Delete</TITLE>
</HEAD>
<BODY bgcolor="<cfoutput>#Application.myBGColor#</cfoutput>">
    <CFIF isDefined("URL.CurIDConnect") AND (isDefined("URL.Name")
        OR isDefined("URL.NameSel"))>
        <CFINCLUDE TEMPLATE = "openftp.cfm">
        <CFIF isDefined("URL.Name")>
            <CFFTP ACTION = "Remove"
                CONNECTION = "FTP"
                ITEM = "#URL.Name#">
        <CFELSE>
            <CFLOOP INDEX = "File" LIST="#URL.NameSel#" >
                <CFFTP ACTION = "Remove"
                    CONNECTION = "FTP"
                    ITEM = "#File#">

```

```
</CFLOOP>
</CFIF>
<CFINCLUDE TEMPLATE = "closeftp.cfm">
</CFIF>
<META HTTP-EQUIV="Refresh"
CONTENT="0;URL=mainftp.cfm?CurIDConnect=<CFOUTPUT>#CurIDConnect#
</CFOUTPUT>">
</BODY>
</HTML>
```

Мы здесь не описываем конфигурационный файл (Application.cfm) для данного приложения в силу того, что он очень схож с конфигурационным файлом для реализации почтовой службы проекта "MyMail", представленного в *приложении 2*. Впрочем, это также касается вспомогательных файлов, таких как main.js, main.css и ie.css. Мы не без оснований назвали представленное нами приложение "упрощенным" FTP-клиентом. Это приложение действительно слишком упрощено. Однако, все в ваших руках!

Резюме

В этой главе мы уделили большое внимание таким популярным протоколам, как HTTP и FTP. Что же касается упомянутого протокола LDAP (Lightweight Directory Access Protocol, упрощенный протокол доступа к каталогам), то он является средством для выполнения основных функций управления сетевой инфраструктурой. Это способствует созданию единого, общего для всей сети, каталога, способного содержать все приложения, сетевые службы и инфраструктуру сети. Централизованные каталоги обеспечат единую стратегию управления сетью, позволяющую устанавливать критерии, в соответствии с которыми будет осуществляться управление сетевой инфраструктурой (например, параметры, задающие время отклика на запрос, или методы аутентификации различных категорий пользователей). Каталог при этом помогает придерживаться заданных критериев управления сетью в масштабах предприятия. В качестве средства для доступа к LDAP-серверу в ColdFusion применяется тег <CFLDAP>. Однако эту тему мы хотим предложить вам изучить самостоятельно.

Подведем итог. В данной главе мы познакомились с CFML-тегами <CFHTTP>, <CFHTTPPARAM> и <CFFTP>.



Организация поисковой системы

При проектировании Web-сайта немаловажно продумать удобный интерфейс с такими возможностями, как иерархическое меню, позволяющее легко ориентироваться в разделах сайта, переход к первой странице при нахождении пользователя на страницах нижнего уровня, карта сайта и т. д.

Однако при создании Web-ресурса, насыщенного разнообразной информацией, представляющей собой запутанную структуру всевозможных документов, наличия вышеперечисленных возможностей будет вряд ли достаточно. И в этой связи было бы уместно в качестве вспомогательного инструмента дополнить Web-сайт системой поиска.

Системы поиска по своей сложности могут быть разными, от простой системы, включающей текстовое поле для ввода критерия поиска с кнопкой выполнения, до самых сложных, использующих дополнительные инструменты для составления гибких запросов и базирующихся на оригинальных алгоритмах поиска.

Поисковый сервер Verity

ColdFusion Server 5 содержит ограниченную OEM-версию сервера Verity K2, являющегося высокомасштабируемым поисковым сервером. Сервер Verity K2 поддерживает одновременную индексацию распределенных архивов базы данных и обрабатывает сотни параллельных запросов пользователей.

Версия сервера Verity K2, являющаяся частью ColdFusion 5, имеет следующие ограничения:

- ☐ в ColdFusion Professional сервер Verity K2 может производить поиск максимум в 125 000 документах;
- ☐ в ColdFusion Enterprise сервер Verity K2 может производить поиск максимум в 250 000 документах.

Поисковый сервер Verity в ColdFusion работает в двух режимах:

- ☐ режим VDK является обычным поисковым режимом Verity;
- ☐ режим K2 представляет собой высокоэффективный режим поиска.

По умолчанию, если вы не сконфигурируете ColdFusion для использования сервера Verity K2, система будет работать в обычном режиме VDK. Verity Server K2 устанавливается по умолчанию с ColdFusion Server, но активизируется вручную путем запуска исполняемого командного файла и является "урезанной" версией. Чтобы использовать режим K2, необходимо отредактировать файл регистрации сервера k2server.ini, сконфигурировать ColdFusion для использования сервера K2 и перезапустить исполняемый файл k2server.exe с соответствующей комбинацией ключей. Для устранения вышеперечисленных ограничений на максимальное количество используемых документов при поиске следует установить полную лицензионную версию сервера Verity Server K2.

Verity Server K2 — высокорезультативный поисковый механизм, разработанный для быстрой обработки запросов в высокоэффективной распределенной системе. Система поиска K2 работает по технологии клиент-сервер. Клиентские приложения K2, такие как приложения ColdFusion, обеспечивают пользователям доступ к индексным документам, хранящимся в коллекциях Verity. Сервер K2 — комплексное приложение, отслеживающее любые изменения, сделанные индексирующими приложениями. Система поиска Verity Server K2 позволяет воспользоваться преимуществом самых последних технологий в области аппаратных средств ЭВМ и программного обеспечения и предоставляет следующие возможности:

- ☐ комплексная архитектура;
- ☐ поддержка функций корректировки знаний Verity, включая тематические разделы;
- ☐ непрерывная операционная поддержка;
- ☐ инкрементное сжатие;
- ☐ хорошие возможности для масштабирования.

Принцип организации поисковой системы

ColdFusion Server предоставляет этапы создания поисковой системы.

- ☐ Создание коллекции или псевдонима на уже существующую коллекцию. Данное действие можно осуществлять как визуально, используя страницу администратора "ColdFusion Administrator" в разделе **Server | Data Sources | Verity Collections** (Сервер | Источники данных | Коллекции для системы Verity), так и программным образом с использованием тега `<CFCOLLECTION>`, который будет описан ниже.
- ☐ Индексирование данных при использовании существующей коллекции. Для индексации данных ColdFusion предоставляет тег `<CFINDEX>`.

- ❑ Многократный поиск по индексированным данным при помощи существующей коллекции или набора коллекций. Поиск в ColdFusion осуществляется с использованием тега <CFSEARCH>.

Коллекции в ColdFusion представляют собой некоторое хранилище на физическом уровне, состоящее из двух каталогов: Custom и File, по структуре подобных друг другу. При этом коллекция регистрируется с уникальным именем, по которому и происходит обращение к данным конкретной коллекции.

Коллекции, в свою очередь, могут быть внутренними, или еще будем их называть "родными" — это коллекции, созданные с помощью средств ColdFusion, и коллекции внешние, сформированные инструментами других приложений. При выполнении поиска не следует использовать вместе "родные" и внешние коллекции.

Большинство существующих классических информационно-поисковых систем применяют механизм автоматического сжатия (индексации) текста документов. Задачей автоматической индексации является составление индекса документа — информации, в сжатом виде представляющей основной смысл документа. Индекс составляется независимо по отношению к отдельному запросу пользователя и рассчитан на многократное применение.

В основе автоматизированных информационно-поисковых систем, обрабатывающих документы определенной тематики, лежат воспроизводящие инженерно-лингвистические модели, ориентированные либо на выделение наиболее содержательных фрагментов обрабатываемого текста (синтаксическое сжатие), либо на опознание, выделение и смысловую переработку той лексики и фразеологии текста, которая передает понятия, соответствующие его тематике (лексическое индексирование, аннотирование, реферирование).

ColdFusion Verity поддерживает широкий спектр типов документов и файлов, которые можно подвергать индексации с последующим поиском и выводом рейтинговых результатов.

Список текстовых форматов, поддерживаемых ColdFusion Verity, таков:

- ❑ Applix (версии 4.2, 4.3, 4.4);
- ❑ HTML;
- ❑ Lotus AmiPro (версия 2.3);
- ❑ Lotus Ami Professional Write Plus;
- ❑ Lotus Word Pro (версии 96, 97);
- ❑ Microsoft RTF;
- ❑ Microsoft Word (версии 2, 6, 95, 97, 2000);

- ☐ Microsoft Word Mac (версии 4, 5, 6);
- ☐ Microsoft Word PC (версии 4, 5, 6);
- ☐ Microsoft Works;
- ☐ Microsoft Write;
- ☐ PDF;
- ☐ текстовые файлы;
- ☐ Unicode;
- ☐ WordPerfect (версии 5.x, 6, 7, 8);
- ☐ WordPerfect Mac (версии 2, 3);
- ☐ HyWrite (версия 4.12).

Список форматов электронных таблиц, поддерживаемых ColdFusion Verity:

- ☐ Applix (версии 4.3, 4.4);
- ☐ Corel QuattroPro (версия 7, 8);
- ☐ Lotus 1-2-3 (версии 2, 3, 4, 5, 96, 97);
- ☐ Microsoft Excel (версии 3, 4, 5, 95, 97, 2000);
- ☐ Microsoft Works.

Список графических форматов, поддерживаемых ColdFusion Verity:

- ☐ Corel Presentations (версии 7.0, 8.0);
- ☐ Lotus Freelance (версии 96, 97);
- ☐ Microsoft PowerPoint (версии 4.0, 95, 97, 2000).

Организация же интерфейса поиска и включения поддержки тех или иных типов файлов в большей степени зависит от требований к вашему Web-ресурсу.

Управление коллекциями

Как уже было отмечено, ColdFusion позволяет создавать коллекции визуально, используя при этом страницу администратора "ColdFusion Administrator" в разделе **Server | Data Sources | Verity Collections** (Сервер | Источники данных | Коллекции для системы Verity). К примеру, для создания коллекции MyLineCruisesCol необходимо в упомянутом разделе страницы администратора ввести наименование коллекции (в поле **Name** (Имя)), месторасположение коллекции (в поле **Path** (Путь)) и выбрать язык, поддерживаемый для данной коллекции (в раскрывающемся списке **Language** (Язык)), как это показано на рис. 19.1.

По умолчанию ColdFusion предлагает размещать новые коллекции в каталоге X:\CFUSION\verity\collections\, где X — устройство, на которое был предустановлен пакет ColdFusion Server.

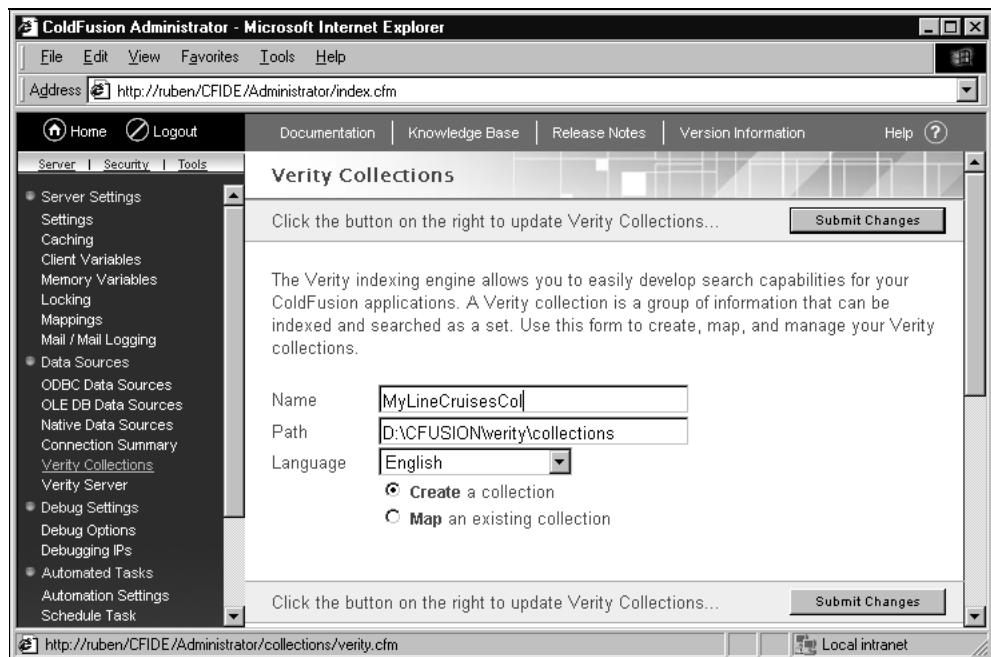


Рис. 19.1. Страница "ColdFusion Administrator", раздел **Verity Collections** для ввода данных

Заполнив данные для создаваемой коллекции, следует воспользоваться кнопкой **Submit Changes** (Применить изменения), что позволит выполнить команду создания коллекции. Далее при успешном выполнении указанной команды созданная коллекция будет отражена в списке **Connected Verity Collections** (Подключенные коллекции для системы Verity), в чем можно убедиться из рис. 19.2.

Здесь же, как видно из рис. 19.2, по отношению к подключенным коллекциям можно совершать различные действия с помощью соответствующих кнопок:

- ☐ **Index** (Индексировать) — производит индексирование текущей коллекции;
- ☐ **Repair** (Восстановить) — восстанавливает испорченные данные в текущей коллекции;
- ☐ **Optimize** (Оптимизировать) — совершает чистку и реорганизует данные для эффективной работы текущей коллекции;

- ❑ **Purge** (Очистить) — очищает данные из текущей коллекции, оставляя коллекцию пригодной для последующего заполнения данными;
- ❑ **Delete** (Удалить) — удаляет выбранную из списка коллекцию.

Впрочем, ColdFusion оснащен инструментом, позволяющим управлять коллекциями на программном уровне. И в качестве такого инструмента представляем тег `<CFCOLLECTION>`.

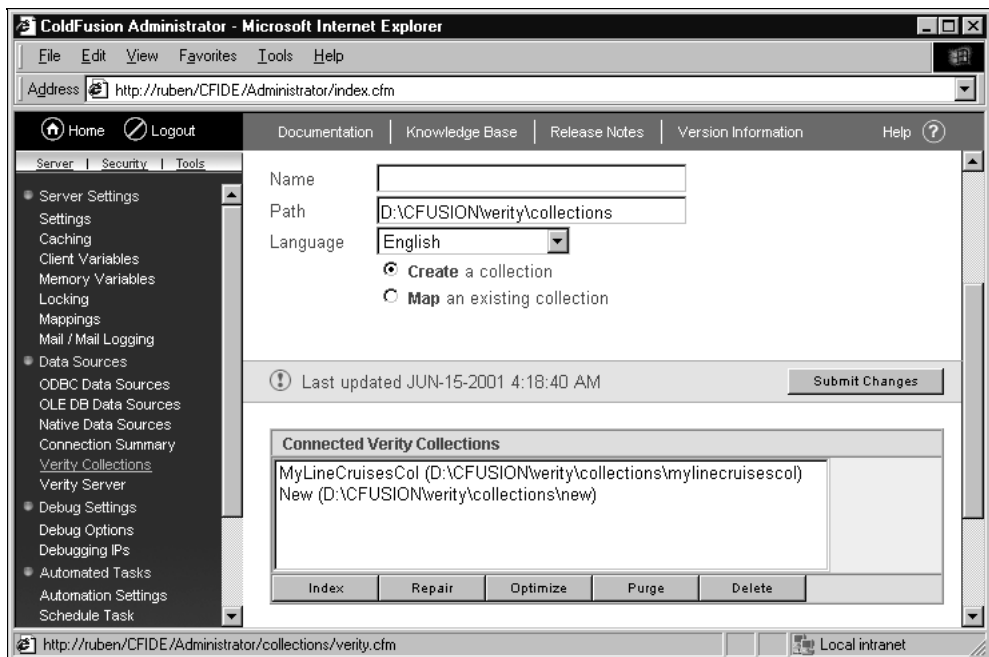


Рис. 19.2. Страница "ColdFusion Administrator", раздел **Verity Collections** отображает результат создания коллекции

Тег `<CFCOLLECTION>`

Синтаксис:

```
<CFCOLLECTION
```

```
    ACTION = "Create | Repair | Delete | Optimize | Map"
```

```
    COLLECTION = "collection_name"
```

```
    PATH = "path_of_verity_directory"
```

```
    LANGUAGE = "language_name">
```

В табл. 19.1 приведено описание атрибутов тега `<CFCOLLECTION>`.

Таблица 19.1. Описание атрибутов тега <CFCOLLECTION>

Атрибут	Описание
ACTION	Действие. Возможные значения: • <code>Create</code> — создает коллекции по указанному пути; • <code>Repair</code> — восстанавливает испорченные данные в существующей коллекции; • <code>Delete</code> — удаляет коллекцию; • <code>Optimize</code> — производит чистку и реорганизует данные для эффективной работы; • <code>Map</code> — назначает псевдоним существующей коллекции. Обязательный атрибут
COLLECTION	Имя коллекции или псевдонима для действия <code>Map</code> . Обязательный атрибут
PATH	Путь к коллекции <code>Verity</code>, где должна быть создана коллекция или псевдоним. Обязательный атрибут для действий <code>Create</code> и <code>Map</code>. При создании коллекции с именем <code>MyFirstCol</code> и указанием пути <code>D:\CFUSION\verity\collections\</code> будет создан каталог <code>D:\CFUSION\Verity\collections\MyFirstCol</code> с необходимыми подкаталогами для данной коллекции. При указании же псевдонима (<code>Map</code>) должен быть определен полный путь к коллекции, например для коллекции <code>MyFirstCol</code> путь может быть определен следующим образом: <code>D:\CFUSION\verity\collections\MyFirstCol</code>
LANGUAGE	Язык, используемый для данной коллекции. Возможные значения: • <code>English</code> — английский язык (по умолчанию); • <code>Danish</code> — датский язык; • <code>Dutch</code> — голландский язык; • <code>Finnish</code> — финский язык; • <code>French</code> — французский язык; • <code>German</code> — немецкий язык; • <code>Italian</code> — итальянский язык; • <code>Norwegian</code> — норвежский язык; • <code>Portuguese</code> — португальский язык; • <code>Spanish</code> — испанский язык; • <code>Swedish</code> — шведский язык. Для использования данного атрибута должна быть установлена международная версия программы поиска — <code>ColdFusion International Pack</code>. Необязательный атрибут

Пример управления коллекциями

В качестве примера приведем код, позволяющий создать уже упомянутую коллекцию MyLineCruisesCol:

```
<CFCOLLECTION
    ACTION = "Create"
    COLLECTION = "MyLineCruisesCol"
    PATH = "D:\CFUSION\Verity\Collections\"
    LANGUAGE = "English">
```

Для удаления созданной нами коллекции достаточно выполнить следующий код:

```
<CFCOLLECTION
    ACTION = "Delete"
    COLLECTION = "MyLineCruisesCol">
```

Аналогичным образом можно совершать описанные выше действия по отношению к любой из существующих коллекций.

Индексирование данных

Не поленимся еще раз отметить, что задачей индексации является составление индекса документа — информации, в сжатом виде доносящей основной смысл документа. Индекс составляется независимо по отношению к отдельному запросу пользователя и рассчитан на многократное применение. В качестве инструмента индексации в ColdFusion используется тег `<CFINDEX>`.

Тег `<CFINDEX>`

Синтаксис:

```
<CFINDEX
    COLLECTION = "collection_name"
    ACTION = "Update | Delete | Purge | Refresh"
    TYPE = "File | Path | Custom"
    TITLE = "title"
    KEY = "ID"
    BODY = "body"
    CUSTOM1 = "custom_value"
    CUSTOM2 = "custom_value"
    URLPATH = "URL"
    EXTENSIONS = "file_extensions">
```

```

QUERY = "query_name"
RECURSE = "Yes | No"
EXTERNAL = "Yes | No"
LANGUAGE = "language_name">

```

В табл. 19.2 представлено описание атрибутов тега `<CFINDEX>`.

Таблица 19.2. Описание атрибутов тега `<CFINDEX>`

Атрибут	Описание
COLLECTION	Наименование коллекции. При индексации внешней коллекции, когда атрибут <code>EXTERNAL="Yes"</code> , необходимо указывать полный путь к коллекции. Обязательный атрибут
ACTION	Действие. Возможные значения: • <code>Update</code> — обновляет индекс с добавлением ключа, указанного атрибутом <code>KEY</code>, если данный ключ еще не определен; • <code>Delete</code> — удаляет ключ из данной коллекции, указанный с помощью атрибута <code>KEY</code>; • <code>Purge</code> — очищает данные из указанной коллекции, оставляя коллекцию пригодной для последующего заполнения данными; • <code>Refresh</code> — очищает данные из указанной коллекции перед заполнением очередными данными. Необязательный атрибут
TYPE	Тип индексации. Возможные значения: • <code>File</code> — файл; • <code>Path</code> — путь; • <code>Custom</code> — пользовательский. Необязательный атрибут
TITLE	Заголовок для коллекции, так же может принимать значение колонки запроса. Обязательный атрибут для типа индексации со значением <code>Custom</code>
KEY	Уникальный идентификатор. Возможные значения: • имя файла, если <code>TYPE="File"</code>; • полный путь, если <code>TYPE="Path"</code>; • наименование колонки таблицы, участвующей в запросе и являющейся уникальным ключом (<code>primary key</code>), или любая другая колонка таблицы из того же запроса, используемая в качестве уникального идентификатора, где атрибут <code>TYPE="Custom"</code>. Необязательный атрибут

Таблица 19.2 (окончание)

Атрибут	Описание
BODY	ASCII-текст для индексирования или колонка, по которой будет произведена индексация, участвующей в запросе таблицы. Обязательный атрибут для <code>TYPE="Custom"</code> . При использовании запроса можно указывать сразу несколько колонок через запятую
CUSTOM1	Пользовательское поле 1, которое будет храниться с результатом индексации. В качестве значения можно использовать колонку запроса. Необязательный атрибут
CUSTOM2	Пользовательское поле 2, аналогично атрибуту <code>CUSTOM1</code>
URLPATH	Путь гиперссылки для файлов. Необязательный атрибут. Можно использовать, когда <code>TYPE="File"</code> или <code>TYPE="Path"</code> . Например: <code>URLPATH = "http://127.0.0.1/my/"</code> . В этом случае при использовании тега <code><CFSEARCH></code> полное значение выводимой колонки <code>URL</code> будет принимать значение: <code>"http://127.0.0.1/my/" + "имя_файла"</code> . Если данный атрибут не указывается, при использовании все того же тега <code><CFSEARCH></code> переменная <code>URL</code> будет принимать значение <code>"имя_файла"</code>
EXTENSIONS	Список расширений рассматриваемых файлов, указанных через запятую. Необязательный атрибут. По умолчанию: <code>htm, html, cfm, cfml, dbm, dbml</code> . Чтобы использовать файлы без расширения, можно включить в список следующее значение: <code>"*."</code>
QUERY	Имя запроса. Позволяет включать в коллекцию данные из БД. Необязательный атрибут
RECURSE	Индикатор рекурсии. Возможные значения: - Yes — да; - No — нет (по умолчанию). При использовании атрибута <code>TYPE="Path"</code> и <code>RECURSE="Yes"</code> индексируются также документы, находящиеся в подкаталогах относительно пути, заданного атрибутом <code>KEY</code>. Необязательный атрибут
EXTERNAL	Индикатор расположения коллекции. Возможные значения: - Yes — да; используется внешняя коллекция, при этом необходимо указывать полный путь к коллекции; - No — нет (по умолчанию); используется внутренняя коллекция ColdFusion, включенная в поисковую систему Verity Необязательный атрибут
LANGUAGE	Аналогичен одноименному атрибуту тега <code><CFCOLLECTION></code>

Примеры индексирования данных

Приведенные ниже примеры подразумевают наличие коллекции `MyLineCruisesCol`. Предположим, что нам необходимо проиндексировать документы, находящиеся в каталоге `C:\Apache\htdocs\MyLineCruises\` и имеющие расширения `htm`, `html`, `cfm`, `cfml`, `doc` и `xls`:

```
<CFINDEX
    COLLECTION = "MyLineCruisesCol"
    ACTION = "Refresh"
    TYPE = "Path"
    EXTENSIONS = ".htm, .html, .cfm, .cfml, .doc, .xls"
    KEY = "C:\Apache\htdocs\MyLineCruises\">
```

Здесь с помощью атрибута `ACTION` вызывается действие `Refresh` (Освежить), позволяющее удалить данные из указанной коллекции и заполнить эту коллекцию новой информацией.

С помощью тега `<CFINDEX>` также можно индексировать данные, полученные на основании пользовательских запросов. Характерным примером может являться использование тега `<CFQUERY>`, позволяющего сформировать запрос к базе данных и тем самым получить объект запроса с набором данных. А уже впоследствии полученный набор данных индексируется согласно необходимому порядку. Приведем пример:

```
<CFQUERY NAME="GetTour" DATASOURCE="Cruise">
    SELECT Key_tou, Name_tou, PointDepart_tou, NumLandingPlaces_tou,
           Time_tou, CostAdult_tou, CostChildren_tou, CostSenior_tou
    FROM Tour
</CFQUERY>
```

```
<CFINDEX
    COLLECTION = "MyLineCruisesCol"
    TITLE = "Name_tou"
    BODY = "PointDepart_tou, NumLandingPlaces_tou"
    ACTION = "Refresh"
    TYPE = "Custom"
    KEY = "Key_tou"
    CUSTOM1 = "Time_tou"
    QUERY = "GetTour">
```

В данном фрагменте кода с помощью атрибута `QUERY` тега `<CFINDEX>` было указано имя предварительно созданного запроса. При этом следует обратить внимание на то, что индексирование производится с пользовательским ти-

пом `TYPE="Custom"`, указан обязательный в этом случае атрибут заголовка `TITLE` и в качестве идентификатора `KEY` выбрано поле таблицы, являющееся уникальным ключом.

Поиск данных

Теперь, когда мы имеем представление о коллекциях и индексации данных, имеет смысл поговорить, для чего, собственно, все это задумано. Для поиска. Поиск в ColdFusion осуществляется с помощью тега `<CFSEARCH>`.

Тег <CFSEARCH>

Синтаксис:

```
<CFSEARCH
    NAME = "search_name"
    COLLECTION = "collection_name"
    TYPE = "Simple | Explicit"
    CRITERIA = "search_expression"
    MAXROWS = "number"
    STARTTROW = "row_number"
    EXTERNAL = "Yes | No"
    LANGUAGE = "language_name">
```

В табл. 19.3 представлено описание атрибутов тега `<CFSEARCH>`.

Таблица 19.3. Описание атрибутов тега `<CFSEARCH>`

Атрибут	Описание
NAME	Имя запроса, включающего результат поиска. Обязательный атрибут
COLLECTION	Имя коллекции. При поиске из внешней коллекции, когда атрибут <code>EXTERNAL="Yes"</code> , необходимо указывать полный путь к коллекции. Можно перечислять несколько коллекций через запятую, однако нельзя сочетать внутренние и внешние коллекции. Обязательный атрибут
TYPE	Тип поиска. Возможные значения: - Simple — простой (по умолчанию); используются операторы¹ <code>STEM</code> и <code>MANY</code>; - Explicit — подробный; все операторы должны быть вызваны явным образом. Необязательный атрибут

¹ См. разд. "Построение выражений поиска" данной главы.

Таблица 19.3 (окончание)

Атрибут	Описание
CRITERIA	Критерий поиска. Необязательный атрибут
MAXROWS	Определяет максимальное количество записей для возвращаемого результата поиска. По умолчанию не ограничивает возвращаемое количество записей. Необязательный атрибут
STARTROW	Номер записи, с которой следует выводить полученные данные. По умолчанию 1. Необязательный атрибут
EXTERNAL	Индикатор расположения коллекции. Возможные значения: • Yes — да; используется внешняя коллекция, при этом необходимо указывать полный путь к коллекции; • No — нет (по умолчанию); используется внутренняя коллекция ColdFusion, включенная в поисковую систему Verity. Необязательный атрибут
LANGUAGE	Аналогично описанию тега <CFCOLLECTION>

При использовании тега <CFSEARCH> создается объект запроса, обеспечивающий доступ к информации относительно результата поиска. При этом в полученный объект запроса включаются следующие колонки:

- ☐ queryName.URL — гиперссылка, определяемая сочетанием указанного атрибута URLPATH тега <CFINDEX> и именем файла, который содержит искомый фрагмент. Данная колонка является пустой при индексации с пользовательским типом, т. е. когда с тегом <CFINDEX> указан атрибут TYPE="Custom";
- ☐ queryName.Key — уникальный идентификатор записи, определяемый при индексации используемой коллекции;
- ☐ queryName.Title — заголовок записи, определяемый при индексации используемой коллекции;
- ☐ queryName.Score — рейтинг результата подсчета уместности найденной записи относительно заданного критерия с помощью атрибута CRITERIA;
- ☐ queryName.Custom1 — пользовательское поле 1, указанное атрибутом CUSTOM1 тега <CFINDEX>;
- ☐ queryName.Custom2 — пользовательское поле 2, указанное атрибутом CUSTOM2 тега <CFINDEX>;
- ☐ queryName.Summary — краткое содержание, автоматически сгенерированное при индексации. По умолчанию выбирает три лучших предложения, соответствующих установленному критерию поиска с суммарным максимальным значением до 500 символов;

- ❑ `queryName.currentRow` — текущая запись запроса при использовании тега `<CFOUTPUT>`.

Также после выполнения тега `<CFSEARCH>` формируются три переменные, позволяющие анализировать созданный объект запроса с результатом поиска. Далее приведем список этих переменных:

- ❑ `queryName.recordCount` — количество записей в данном запросе (удовлетворяющих критерию поиска);
- ❑ `queryName.columnList` — список колонок запроса, перечисленных через запятую;
- ❑ `queryName.recordsSearched` — число документов, подвергшихся поиску или перебору.

Пример поиска

Мы уже неоднократно возвращались к реализованному нами проекту "My-Line Cruises". И этот раздел не будет исключением. Далее в листинге 19.1 представлен код страницы `Find.cfm`, которая позволяет выводить результат поиска в соответствии с используемым критерием и рейтинговым показателем по каждой найденной записи.

Листинг 19.1. Код страницы `Find.cfm`

```
<HTML>
<HEAD>
  <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
  <link rel="STYLESHEET" type="text/css" href="main.css">
  <script type="text/javascript" language="JavaScript"
    src="main.js"></script>
  <script language="JavaScript">
    if (ie) {document.write('<link rel="stylesheet" type="text/css"
      href="main_ie.css" title="master">');}
  </script>
</HEAD>
<BODY onload="init(1)"
  bgcolor="#<CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
  text="Black" alink="#107098" link="#0C1C47" vlink="#003366">

  <CFINCLUDE TEMPLATE = "BasicBlock.cfm">

  <DIV id="text1">
    <CFIF isDefined("URL.Criterion")>
```

```

<CFSEARCH NAME = "MySearch"
      COLLECTION = "MyLineCruisesCol"
      TYPE="Explicit"
      CRITERIA = "#URL.Criterion#">

<CFOUTPUT>

<TABLE width="600" border='0' cellspacing='4' cellpadding='4'>
  <TR>
    <TD class="title">Search Results: "#URL.Criterion#"</TD>
  </TR>
  <TR>
    <TD class="menleft">
      <CFIF MySearch.RecordCount eq 0>
        The records by your criterion of search are absent.
      <CFELSE>
        In total is found #MySearch.RecordCount#
        of #MySearch.RecordsSearched# record(s).
      </CFIF>
    </TD>
  </TR>
</TABLE>
</CFOUTPUT>

<TABLE width="600" border='0' cellspacing='4' cellpadding='4'>
  <TR align="center">
    <TD class="txtTitleTab" width="20">Row</TD>
    <TD class="txtTitleTab" width="50">Score</TD>
    <TD class="txtTitleTab" width="100">Title</TD>
    <TD class="txtTitleTab" width="430" align="left">Summary</TD>
  </TR>
  <CFOUTPUT QUERY="MySearch">
    <TR><TD colspan="4" bgcolor="122c6d" height=1></td></TR>
    <TR valign="top">
      <TD class="txtTab" align="right">#CurrentRow#</TD>
      <TD class="txtTab" align="right">#Score#</TD>
      <TD class="txtTab">
        <a href = "#URL#" target = "blank">#Title#</a></TD>
      <TD class="txtTab">#Summary#</TD>
    </TR>

```

```

        </CFOUTPUT>
    </TABLE>
<CFELSE>
    <CFOUTPUT>
        <p class="txtError">
            The criterion of search is absent!
        </p>
    </CFOUTPUT>
</CFIF>
</DIV>
</BODY>
</HTML>

```

Перед использованием рассмотренного выше кода данные были проиндексированы в соответствии с первым примером индексации, приведенным в этой главе. На рис. 19.3 представлен результат поиска по критерию "Cruise" уже после индексации данных.

My-Line Cruises - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://127.0.0.1/MyLineCruises/Find.cfm?criterion=Cruise>

Site Finder

My-Line Cruises

« Cruises Schedules Guidance Buy Tickets City Map Contact Us

Search Results: "Cruise"

In total is found 3 of 31 record(s).

Row	Score	Title	Summary
1	0.9017	Circle Line	The Seaport Liberty Cruise Schedule Pier 16, South Street Seaport March 24 - April 27 Seaport Liberty 1 hour Mon. - Fri.: 12pm, 1:30pm, 3pm, 4:30pm Sat & Sun: 12pm, 1:30pm, 3pm, 4:30pm, 6pm April 28 - October 28 Seaport Liberty 1 hour Mon. - Fri.: 12pm, 1:30pm, 3pm, 4:30pm Sat & Sun: 12pm, 1:30pm, 3pm, 4:30pm, 6pm, 7:30pm November 1 - December 23 Thursday - Sunday Seaport Liberty 1 hour 12pm, 1:30pm, 3pm, 4:30pm Closed Thanksgiving Day. Prices Seaport Liberty 1 hour Adults \$13, Seniors \$11, ...
2	0.8510	Sightseeing Cruises	Sightseeing Cruises See the City from a Unique Perspective Sightseeing Cruises are available year-round from Pier 83 at 42nd Street on the Hudson River, and from March through December from the South Street Seaport, Pier 16 1-hr Seaport Liberty Seaport Liberty Cruise Pier 16, South Street Seaport "One Skyline. On our delightful Seaport Liberty Cruise, you'll get a great view of the downtown skyline, Ellis Island and Lady Liberty. 2-hr Semi Circle The Semi-Circle Cruise Pier 83, West 42nd ...
3	0.7742		--- BGColor bec0d1, 89869b --->

Internet

Рис. 19.3. Результат поиска по критерию "Cruise" в рамках проекта "My-Line Cruises"

Между тем, в ColdFusion Studio среди богатого набора мастеров (wizards) существует мастер Verity — Verity Wizard, позволяющий создать на лету набор CFML-страниц для реализации поисковой системы.

При формировании критериев поиска ColdFusion предоставляет большой набор дополнительных операторов, позволяющих добиваться "изысканных" результатов. О чем, собственно говоря, и пойдет речь в следующем разделе.

Построение выражений поиска

При использовании тега <CFSEARCH> можно задавать как простой (Simple) так подробный (Explicit) тип поиска. Что касается простого типа поиска, установленного по умолчанию, то при этом уже подразумевается наличие операторов STEM (ищет всевозможные варианты указанного слова, однокоренные слова) и MANY (подсчитывает плотность слов или фраз, определяя рейтинговую величину для найденных записей). Что же касается подробного поиска, то операторы должны быть вызваны явным образом.

Отметим, что при формировании выражений для отображения специальных символов, как обычных, следует перед ними использовать левую наклонную черту \. Например, для отображения знака квадратной скобки ([]) следует писать:

```
\[
```

Специальными символами следует считать:

```
, ( ) [ ] " \ @ { } < > ' "
```

Сервер ColdFusion позволяет комбинировать операторы и модификаторы, которые необходимо заключать в угловые скобки, например:

```
<STEM>Seaport Liberty Cruise
```

Однако такие операторы как AND (И), OR (ИЛИ) или NOT (НЕТ) можно использовать без скобок. Например:

```
Cruise OR Line NOT Yankee Stadium
```

Для указания приоритета можно добавить круглые скобки. Например:

```
Cruises AND (Liberty OR Harlem)
```

Впрочем, это же условие поиска можно задать еще двумя способами в соответствии с правилом построения выражения:

```
AND (Cruises, Liberty OR Harlem)
```

или

AND (Cruises, OR (Liberty, Harlem))

Поисковый сервер Verity позволяет использовать специальные символы подстановки: * и ?. Здесь символ звездочки (*) подразумевает неопределенное количество любых символов, а знак вопроса (?) предполагает один произвольный символ. Например:

The best developer of the programs is M?cro*

Для данного выражения допустимыми будут и Microsoft, и Macromedia, и Micrografx, и т. д.

Мы уже затронули такие операторы, как AND, OR и NOT. Впрочем, в ColdFusion Verity существует еще ряд операторов, позволяющих выразить логику, которую необходимо применить к элементам поиска. Список возможных операторов таков:

- ❑ STEM — позволяет искать различные варианты указанного слова;
- ❑ WILDCARD — предоставляет возможность использовать шаблоны;
- ❑ WORD — подразумевает поиск строго определенного слова;
- ❑ THESAURUS — позволяет найти вместе с указанным словом также его синонимы;
- ❑ SOUNDEX — позволяет найти вместе с указанным словом созвучные ему слова;
- ❑ TYPO/*N* — находит вместе с указанным словом подобные слова с некоторой погрешностью, где уровень погрешности определяется необязательной переменной *N*. По умолчанию уровень погрешности равен 2. Например: <TYPO/1>Liberty;
- ❑ NEAR — ищет указанные слова, расположенные по тексту рядом в произвольной форме в пределах 1000 слов друг от друга. Уровень близости увеличивает рейтинг. Например: Cruises<NEAR>Line;
- ❑ NEAR/*N* — аналогичен предыдущему с той лишь разницей, что в качестве переменной *N* необходимо указывать максимальное расстояние между словами от 1 до 1024;
- ❑ PARAGRAPH — поиск сочетания перечисленных слов или фраз, находящихся в одном параграфе. Например: <PARAGRAPH>(Liberty, State, Building);
- ❑ PHRASE — поиск по заданной фразе. Например: <PHRASE>(Brooklyn Bridge);
- ❑ SENTENCE — поиск сочетания перечисленных слов или фраз, находящихся в одном предложении. Например: <SENTENCE>(Ellis Island, Wall Street);

- ❑ **IN** — поиск сочетания слов, находящихся в одной определенной зоне. Зона документа определяется квалификатором **WHEN**;
- ❑ **CONTAINS** — осуществляет поиск указанного слова или фразы в заголовках определяемого поля или набора полей (см. ниже). Например: `CF_TITLE<CONTAINS>Line`;
- ❑ **MATCHES** — осуществляет поиск указанного слова или фразы, точно соответствующих содержанию определяемого поля или набора полей (см. ниже). Например: `CF_URL<MATCHES>Commercial.cfm`;
- ❑ **STARTS** — осуществляет поиск указанного слова или фразы в начале определяемого поля или набора полей (см. ниже). Например: `CF_TITLE, CF_URL<STARTS>My`;
- ❑ **ENDS** — осуществляет поиск указанного слова или фразы в конце определяемого поля или набора полей (см. ниже);
- ❑ **SUBSTRING** — осуществляет поиск указанного слова или фразы в подзаголовках определяемого поля или набора полей (см. ниже);
- ❑ **AND** — логический оператор **И**; предполагает поиск всех указанных слов;
- ❑ **OR** — логический оператор **ИЛИ**; предполагает поиск либо одного, либо другого слова;
- ❑ **NOT** — отрицание; используется с операторами **AND** и **OR**;
- ❑ **ACCURUE** — осуществляет поиск, по крайней мере, по одному из указанных слов или фраз. Например: `<ACCURUE>(2-hr, 1:30pm, November)`;
- ❑ **ALL** — осуществляет поиск по всем словам или фразам, которые определены, рейтинг при этом для всех найденных записей назначается 1,00;
- ❑ **ANY** — осуществляет поиск, по крайней мере, по одному из указанных слов или фраз, рейтинг при этом для всех найденных записей назначается 1,00;
- ❑ **YESNO** — позволяет округлять рейтинг до 1;
- ❑ **PRODUCT** — перемножает рейтинги для элементов в каждой записи, отвечающей запросу. Например: `<PRODUCT>(Line, Roosevelt Island)`;
- ❑ **SUM** — складывает рейтинги для элементов в каждой записи, отвечающей запросу, не превышая при этом максимальное значение, равное 1;
- ❑ **COMPLEMENT** — рассчитывает рейтинги записей, отвечающих запросу, путем вычитания из 1 рейтинга найденной записи;
- ❑ **CASE** — задает чувствительность к регистру. Например: `<CASE>LINE`;
- ❑ **MANY** — подсчитывает плотность слов или фраз, определяя рейтинговую величину для найденных записей;
- ❑ **ORDER** — производит сортировку найденных записей в порядке перечисления в запросе. Данный оператор может быть использован в сочетании

с операторами PARAGRAPH, SENTENCE и NEAR/N. Например:
<ORDER><NEAR>(Cruises, Line).

В качестве определяемых полей могут рассматриваться следующие переменные:

- ❑ CF_CUSTOM1 — пользовательское поле 1 индексированной записи;
- ❑ CF_CUSTOM2 — пользовательское поле 2 индексированной записи;
- ❑ CF_KEY — уникальный идентификатор индексированной записи;
- ❑ CF_TITLE — заголовок индексированной записи;
- ❑ CF_URL — гипертекст индексированной записи.

Также для числовых выражений могут быть использованы следующие операторы:

- ❑ = — равно;
- ❑ > — больше;
- ❑ >= — больше или равно;
- ❑ < — меньше;
- ❑ <= — меньше или равно.

Использование Verity Server K2

В начале данной главы мы отметили, что поисковый сервер Verity в ColdFusion работает в двух режимах: в обычном режиме VDK, используемом по умолчанию, и режиме K2, являющемся высокоэффективным режимом поиска. Напрашивается вопрос: "Как же ColdFusion определяет режим работы поискового сервера?".

ColdFusion определяет режим поиска в Verity, сравнивая название коллекции, указанное тегом <CFSEARCH>, со значениями в локальном регистрационном журнале. Если название коллекции найдено, то будет производиться обычный поиск VDK. Имена коллекций заносятся в журнал путем вызова тега <CFCOLLECTION> или с помощью страницы администратора, как уже было рассмотрено выше. Если название коллекции не найдено, ColdFusion использует для поиска режим K2.

Для конфигурирования сервера Verity K2 можно воспользоваться страницей администратора "ColdFusion Administrator", выбрав при этом раздел **Server | Data Sources | Verity Server** (Сервер | Источники данных | Сервер для системы Verity), как это показано на рис. 19.4.

Следующим шагом необходимо отредактировать файл k2server.ini, позволяющий задавать коллекции для поиска с помощью сервера Verity K2. В листинге 19.2 приведен пример конфигурационного ini-файла.

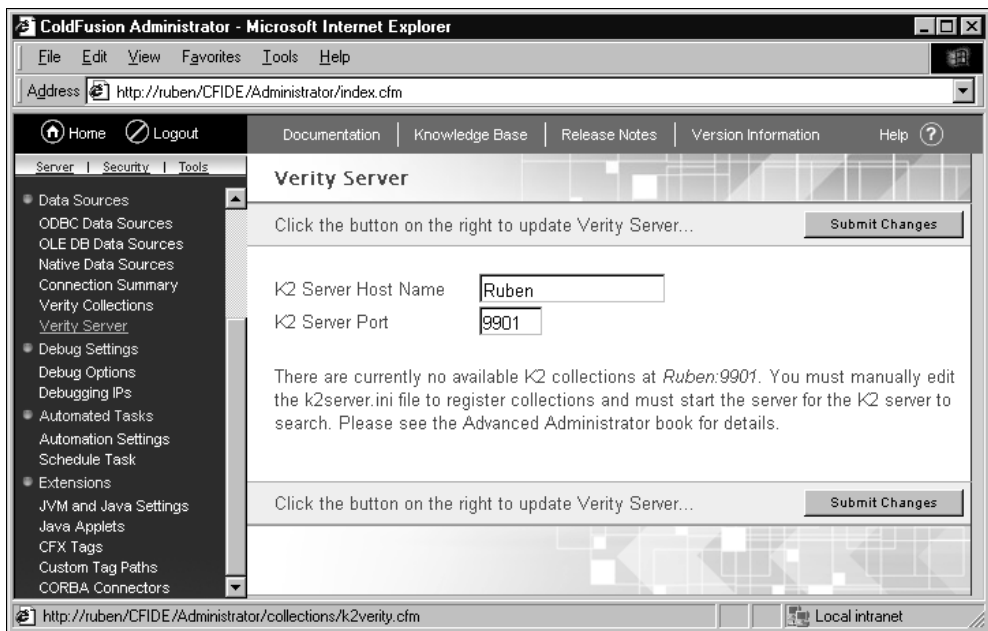


Рис. 19.4. Страница "ColdFusion Administrator", раздел **Verity Server**

Листинг 19.2. Конфигурационный файл k2server.ini сервера Verity K2

```
1 ## Пример ini-файла сервера K2, используемого с ColdFusion.
2
3 [Server]
4
5 ## Число одновременно существующих потоков при поиске данных
6 numThreads=5
7 ## Максимальное число обрабатываемых файлов в одном потоке
8 maxFiles=100
9 ## Максимальное число клиентов, одновременно
10 ## подсоединяющихся к серверу K2
11 numListeners=20
12 ## Номер порта
13 portNo=9901
14 ## Расположение домашнего каталога, используемого
15 ## в качестве ресурсов сервера Verity K2
16 vdkHome=d:\cfusion\verity\common
17
```



```
18 ## Дополнительные параметры, используемые по умолчанию
19 sortTruncDocs=
20 accessProfile=
21 knowledgeBase=
22 charMap=
23 language=
24 locale=
25
26 ## Коллекции
27 [Coll-0]
28 collPath=d:\cfusion\verity\collections\MyLineCruisesCol\file
29 collAlias= MyLineCruisesCol_file
30 topicSet=
31 knowledgeBase=
32 onLine=2
33
34 [Coll-1]
35 collPath=d:\cfusion\verity\collections\MyLineCruisesCol\custom
36 collAlias=MyLineCruisesCol_custom
37 topicSet=
38 knowledgeBase=
39 onLine=2
40 ## Окончание файла
```

ColdFusion использует режим K2 для поиска в коллекциях при условии, что:

- ☐ сервер K2 уже запущен;
- ☐ имя коллекции при использовании тега `<CFSEARCH>` было определено в файле `k2server.ini` и является уникальным, т. е. не используется в коллекциях Verity.

После того как `ini`-файл подготовлен, можно запускать поисковый сервер Verity K2.

В Windows NT для этого следует запустить файл `k2server.exe` с соответствующими параметрами:

```
k2server -ntservice 1 -inifile k2server.ini
```

Теперь, когда поисковый сервер K2 запущен, можно осуществлять поиск. Например:

```
<CFSEARCH NAME = "MySearchK2"
          COLLECTION = "MyLineCruisesCol_file"
```

```
TYPE="Explicit"
```

```
CRITERIA = "<PARAGRAPH>(Liberty, State, Building)">
```

```
<CFTABLE QUERY = "MySearchK2" COLHEADERS>
```

```
<CFCOL HEADER = "Score" TEXT = "#Score#">
```

```
<CFCOL HEADER = "Title" TEXT = "#Title#">
```

```
<CFCOL HEADER = "Summary" TEXT = "#Summary#">
```

```
<CFCOL HEADER = "Custom1" TEXT = "#Custom1#">
```

```
</CFTABLE>
```

Для завершения работы поискового сервера в Windows NT необходимо из командной строки запустить файл k2server.exe следующим образом:

```
k2server -ntservice 0
```

Резюме

Подведем итог. В данной главе мы познакомились с CFML-тегами `<CFCOLLECTION>`, `<CFINDEX>` и `<CFSEARCH>`.



Язык CFScript

До сих пор при формировании того или иного кода мы применяли набор тегов с целью продемонстрировать возможности языка CFML. Однако все тот же ColdFusion позволяет писать программы, используя встроенный язык CFScript, по своему синтаксису похожий на JavaScript. При этом CFScript позволяет заменять "тяжелые" конструкции, написанные с помощью языка CFML. Для применения же CFScript достаточно расположить необходимый код внутри тега `<CFSCRIPT>`.

Тег `<CFSCRIPT>`

Синтаксис:

```
<CFSCRIPT>
```

```
    Код CFScript
```

```
</CFSCRIPT>
```

Например, объявление одномерного массива с помощью CFML-тегов можно записать так:

```
<CFSET CrArray = ArrayNew(1)>
```

```
<CFSET CrArray[1] = "Harlem">
```

```
<CFSET CrArray[2] = "Brooklyn Br.">
```

При использовании CFScript ту же задачу можно реализовать следующим образом:

```
<CFSCRIPT>
```

```
    CrArray = ArrayNew(1);
```

```
    CrArray[1] = "Harlem";
```

```
    CrArray[2] = "Brooklyn Br.";
```

```
</CFSCRIPT>
```

Обратите внимание, что ColdFusion позволяет легко сочетать CFML-теги и CFScript-сценарии. К примеру, несколько усложним представленный выше код и сравним варианты фрагментов программ с использованием CFML-тегов и CFScript (табл. 20.1).

Таблица 20.1. Применения CFML-тегов и CFScript

Вариант программы	Код
С использованием CFML-тегов	<pre> <CFSET Title = "My-Line Cruises"> <CFSET CrArray = ArrayNew(1)> <CFSET CrArray[1] = "Harlem"> <CFSET CrArray[2] = "Brooklyn Br."> <CFOUTPUT> #Title#
 </CFOUTPUT> </pre>
С использованием CFScript	<pre> <CFSCRIPT> Title = "My-Line Cruises"; CrArray = ArrayNew(1); CrArray[1] = "Harlem"; CrArray[2] = "Brooklyn Br."; WriteOutput(Title & "
"); </CFSCRIPT> </pre>
Вывод на экран списка элементов независимо от способа объявления массива	<pre> <CFIF IsArray(CrArray) AND ArrayIsEmpty(CrArray) eq false> <CFLOOP INDEX = "i" FROM = "1" TO = #ArrayLen(CrArray)#> <CFOUTPUT>#i#. #CrArray[i]#</CFOUTPUT>
 </CFLOOP> </pre>

Общие сведения о языке CFScript

CFScript поддерживает все функции и операторы, используемые в CFML. Если справочник функций CFML представлен в *приложении 3*, то операторы CFML подробно рассмотрены в *главе 8*. Так же CFScript поддерживает любые типы переменных ColdFusion, рассмотренные в *главе 9*.

При использовании CFScript можно применять две разновидности комментариев: *однострочные* и *многострочные*. Комментарии, занимающие одну строку, начинаются, как в Java или JavaScript, с символов `//` и заканчиваются в конце строки. Такой стиль удобен для размещения кратких пояснений к отдельным строкам кода. Для более подробных пояснений можно воспользоваться комментариями, расположенными на нескольких строках, начав текст комментариев символами `/*` и закончив символами `*/`. При этом весь текст между указанными парами символов будет расценен как комментарий.

Например:

```
<CFSCRIPT>
    /*
    Преобразование температуры воздуха
    из градусов по Фаренгейту (F) в градусы по Цельсию (C)
    */
    Value = 70;           // Значение температуры по Фаренгейту (F)
    Val = (Value-32)*5/9;
    Measurement = " C";
    Result = NumberFormat(Val, "____.") & Measurement;
    WriteOutput(Result);  // Результат температуры по Цельсию (C)
</CFSCRIPT>
```

В приведенном примере использована одна из функций CFML — `NumberFormat`, позволяющая форматировать числовые величины по заданной маске.

Отметим, что все операторы в CFScript заканчиваются точкой с запятой (;), разрывы строк в коде игнорируются, и CFScript нечувствителен к регистру при выполнении сценария.

Так же, как и в JavaScript, CFScript позволяет использовать собственные функции, по своему синтаксису идентичные JavaScript. Определение функции начинается с ключевого слова `function`, далее следует имя функции, и вся конструкция заканчивается круглыми скобками с необязательным перечислением передаваемых параметров внутри этих скобок, а блок кода функции определяется фигурными скобками.

Синтаксис определения функции таков:

```
function MyFunctionName([paramName1[, paramName2...]]) {
    Тело функции
}
```

Например:

```
<CFSCRIPT>
function Temperature(Value, CtoF) {
    if (isNumeric(Value) and isBoolean(CtoF)) {
        if (YesNoFormat(CtoF) EQUAL "Yes") {
            Val = Value*1.8+32;
            Measurement = " F";
        } else {
            Val = (Value-32)*5/9;
```

```

        Measurement = " C";
    }
    Result = NumberFormat(Val, "____.____") & Measurement;
} else {
    Result = "Error";
}
return Result;
}

WriteOutput("Temperature: " & Temperature(12,"Yes"));
</CFSCRIPT>

```

В результате выполнения приведенного примера с помощью функции `WriteOutput` будет выведена надпись "Temperature:" вместе с результатом работы указанной функции в соответствии с передаваемыми параметрами.

Причем в данном примере мы использовали такие операторы CFScript, как `if...else` и `return`. Оператор `if...else` позволяет в зависимости от некоторого логического значения производить те или иные действия (раздел `else` необязателен). В качестве логического выражения можно указывать выражение с операторами отношения. Это позволяет выводить ту или иную ветвь кода в зависимости от результатов сравнения.

Оператор `return` используется только внутри блока функций и предназначен для немедленного завершения работы функции и передачи управления коду, вызывающему эту функцию. Оператор `return` может записываться с необязательной переменной в качестве возвращаемого значения так, как это показано в нашем примере.

Помимо уже упомянутых операторов `if...else` и `return` в CFScript также существуют следующие операторы: `while`, `do...while`, `for`, `for...in`, `break`, `continue`, `switch/case` и `var`. Давайте познакомимся с каждым из них.

❑ Оператор `while` представляет цикл многократного выполнения до тех пор, пока значение логического выражения равно `True` (истина). Ниже приведен пример для вывода таблицы соответствия температуры воздуха по Цельсию (C) и по Фаренгейту (F) с возможностью определения границы вывода с помощью оператора `while`:

```

<CFSCRIPT>

WriteOutput('<table><tr><td>C</td><td>F</td></tr>');
c = 10;
while ( c LT 40) {
    f = c*1.8+32;
    WriteOutput('<tr><td>' & c & '</td><td>' & f & '</td></tr>');
}

```

```
c = c + 10;  
}  
WriteOutput('</table>');  
</CFSCRIPT>
```

В результате выполнения данного кода будет выведена следующая информация:

```
C  F  
10 50  
20 68  
30 86
```

- ❑ Оператор `do...while` является разновидностью оператора цикла, но в отличие от предыдущего оператора проверяет условие завершения в конце очередного прохода по телу цикла, а не перед ним. Для вывода информации, полученной в результате выполнения предыдущего примера, следует выполнить код:

```
<CFSCRIPT>  
WriteOutput('<table><tr><td>C</td><td>F</td></tr>');  
c = 10;  
do {  
    f = c*1.8+32;  
    WriteOutput('<tr><td>' & c & '</td><td>' & f & '</td></tr>');  
    c = c + 10;  
} while ( c LT 40);  
WriteOutput('</table>');  
</CFSCRIPT>
```

- ❑ Оператор `for` также является разновидностью оператора цикла. В нем предусмотрено определение инициализации переменной, условия завершения цикла и код, выполняемый при каждой итерации, чаще всего используемый для наращивания переменной. Перечисленные части оператора `for` указываются через точку с запятой. Например:

```
<CFSCRIPT>  
for (i=10; i LTE 30; i=i+5) {  
    WriteOutput(i & "<br>");  
}  
</CFSCRIPT>
```

В результате выполнения данного кода будет выведен список значений переменной `i` от 10 до 30 с шагом 5.

- ❑ Оператор `for...in` тоже является разновидностью оператора цикла и позволяет выполнять цикл, перебирая значения *переменной структуры* (structures). В представленном ниже примере после создания структуры выводятся значения ее элементов с использованием оператора цикла `for...in`:

```
<CFSCRIPT>
```

```
TravelerStruct = StructNew();  
TravelerStruct.RS1_98 = "Nomar Garciaparra";  
TravelerStruct.RS2_98 = "Pedro Martinez";  
  
for (trvl in TravelerStruct) {  
    WriteOutput(trvl & " " & TravelerStruct[trvl] & "<br>");  
}
```

```
</CFSCRIPT>
```

В результате выполнения данного кода будет выведена следующая информация:

```
RS1_98 Nomar Garciaparra  
RS2_98 Pedro Martinez
```

- ❑ Оператор `break` сообщает исполняющей среде, что следует прекратить выполнение текущего блока и передать управление оператору, следующему за данным блоком. Например:

```
<CFSCRIPT>
```

```
for (i=10; i LTE 30; i=i+5) {  
    WriteOutput(i & "<br>");  
    if (i EQ 20) {  
        break;  
    }  
}
```

```
</CFSCRIPT>
```

Несмотря на то, что в приведенном выше примере завершение цикла предполагается после того, как переменная `i` достигнет значения 30, оператор `break` прекратит выполнение цикла при указанном условии (в данном примере при значении переменной `i`, равном 20).

- ❑ Оператор `continue`. Иногда возникает ситуация, когда необходимо досрочно перейти к выполнению следующей итерации, проигнорировав часть операторов тела цикла, еще не выполненных в текущей итерации. Для этого следует использовать оператор `continue`, совершающий пере-

ход из текущей точки в конец цикла. При использовании операторов цикла `while` и `do...while` оператор `continue` приводит к пропуску операторов тела цикла, не выполненных на данной итерации, и к передаче управления на проверку условия завершения цикла. С использованием оператора цикла `for` оператор `continue` приводит к тому, что управление передается третьей части определения цикла, т. е. коду, который выполняется на каждой итерации после операторов тела цикла. Например:

```
<CFSCRIPT>
```

```
for (i=10; i LTE 30; i=i+5) {  
    if (i EQ 20) {  
        continue;  
    }  
    WriteOutput(i & "<br>");  
}
```

```
</CFSCRIPT>
```

В результате выполнения данного кода будет выведен список значений переменной `i` от 10 до 30 с шагом 5 с пропуском значения 20.

- ❑ Оператор `switch/case` — оператор-переключатель, обеспечивающий переключение между различными частями программного кода в зависимости от значения некоторой переменной или выражения. Применение данного оператора целесообразно, когда одна переменная может принимать более двух различных значений, и для каждого принимаемого значения этой переменной необходимо совершить то или иное действие, реализованное посредством программного кода. Например:

```
<CFSCRIPT>
```

```
switch (DayOfWeek(now())) {  
    case 2: { // Monday  
        DinnerPlace = "Vong";  
        break;  
    }  
    case 3: { // Tuesday  
        DinnerPlace = "San Domenico";  
        break;  
    }  
    case 4: { // Wednesday  
        DinnerPlace = "Felidia";  
        break;  
    }  
}
```

```
case 5: { // Thursday
    DinnerPlace = "Le Cirque";
    break;
}
case 6: { // Friday
    DinnerPlace = "Russian Tea Room";
    break;
}
default: { // Free days
    DinnerPlace = "Home";
    break;
}
}

WriteOutput("Today I have dinner in: " & DinnerPlace);
</CFSCRIPT>
```

Представленный пример позволяет выводить определенную информацию в зависимости от текущего дня недели. Заметим, что синтаксис оператора `switch/case` позволяет определять программный код для нерассмотренных вариантов, используя при этом необязательный блок `default`, как показано в приведенном выше примере.

- ❑ Оператор `var` позволяет создавать и инициализировать переменные. Так же, как и оператор `return`, оператор `var` необходимо использовать только внутри пользовательских функций. Важным является следующий момент: объявление переменных с использованием оператора `var` следует выполнять "наверху", т. е. перед указанием любых других операторов внутри функции. И еще. Все переменные, созданные с помощью оператора `var`, являются видимыми только в функции, в которой создаются, в отличие от переменных, объявленных без использования оператора `var`. Например:

```
<CFSCRIPT>

function Multiplication(x, y) {
    var Result = x * y;
    return Result;
}

WriteOutput("x * y = " & Multiplication(11, 7));

</CFSCRIPT>
```

В CFScript, как и в любом другом языке, существует понятие зарезервированных ключевых слов. Эти ключевые слова совместно с синтаксисом опе-

раторов и разделителей входят в описание языка CFScript. Они могут применяться только по назначению, их нельзя использовать в качестве идентификаторов для имен переменных или функций. Как уже можно догадаться, в дополнение к наименованиям функций и операторов языка CFML также в список зарезервированных слов языка CFScript входят: `if`, `else`, `return`, `do`, `while`, `for`, `in`, `break`, `continue`, `switch`, `case`, `default`, `var`.

Заметим, что внутри блока `<CFSCRIPT>` не следует использовать CFML-теги. И еще: в отличие от JavaScript, в CFScript недопустимо обращение к некоторым неявным объектам, таким как `window` и `document`.



Тег <CFOBJECT> и технологии COM, CORBA и EJB

Прежде чем описывать работу с тегом <CFOBJECT>, позволяющим вызывать распределенные объекты, построенные на основе таких технологий, как COM, CORBA и EJB, приведем некоторые пояснения к каждой из технологий.

COM

COM (Component Object Model, модель многокомпонентных объектов) представляет собой "фундамент", на котором построены технологии ActiveX и OLE. Технология COM создавалась фирмой Microsoft как средство взаимодействия приложений (в том числе составных частей операционной системы) Windows, функционирующих на одном компьютере, с последующим развитием для использования в пределах локальной сети. Главная задача на момент создания — обеспечение технологии Object Linking and Embedding (OLE 1.0). Характерно, что обмен данными между приложениями (Dynamic Data Exchange, DDE) первоначально строился не по COM-технологии, а с использованием механизма сообщений (messages). Технология развивается по мере добавления новых возможностей. Как универсальная технология взаимодействия приложений, она начала использоваться с OLE 2.0 (1991 г.). Концепция технологии неразрывно связана с ее реализацией. Возникновение новых возможностей — это просто появление новых библиотек, функций API и утилит Windows. "Общий знаменатель" технологии — двоичная структура объекта, хотя в настоящий момент существует язык описания структуры объекта — Interface Definition Language (IDL).

COM содержит все необходимое для построения распределенной системы:

- ☐ технологию удаленного вызова методов как статических, так и динамических;
- ☐ базы данных серверных объектов (библиотеки типов), которые могут быть импортированы для анализа структуры серверов COM;
- ☐ универсальный протокол обмена между клиентами и серверами;
- ☐ спецификации так называемых "составных документов" (ActiveDoc);

□ объектный монитор транзакций (MTS), компонентную модель (ActiveX) и др.

Все составные части прекрасно соответствуют друг другу в рамках модели COM. Уникальной возможностью COM является универсальная технология доступа к базам данных — OLE ADO DB.

COM реализует высокий уровень абстракции — все вопросы низкого уровня, такие как взаимодействие с операционной системой или сетевыми средствами, скрыты от прикладного программиста. Передача данных между клиентом и сервером основана на наборе типов OLE Automation.

CORBA

CORBA (Common Object Request Broker Architecture, универсальная архитектура посредничества при запросе объектов) создавалась консорциумом OMG (Object Management Group). OMG представляет собой некоммерческую организацию, являющуюся содружеством разработчиков программного обеспечения и его потребителей, объединивших свои усилия для создания спецификаций этой технологии. В настоящий момент OMG насчитывает более 800 членов, включая всех сколько-нибудь серьезных производителей программного обеспечения. Первая спецификация CORBA появилась в 1991 г. Новые возможности официально считаются добавленными в CORBA в момент утверждения соответствующей спецификации. Как правило, в разработке спецификации участвуют крупнейшие специалисты в данной области. Разработка реализации — задача конкретной фирмы. Обычно от утверждения спецификации до появления высококачественной реализации проходит довольно много времени — иногда несколько лет. "Общий знаменатель" технологии — объявления на языке IDL, который является "сердцем" CORBA с момента ее появления. (Существуют три различных языка описаний с одним и тем же названием — OSF IDL, Microsoft IDL и OMG IDL.)

CORBA так же, как и COM, поддерживает статический и динамический способ организации удаленных вызовов и обеспечивает высокий уровень абстракции за счет базирования технологии исключительно на языке описания IDL с последующей трансляцией таких спецификаций на конкретный язык программирования.

EJB и Java

EJB (Enterprise Java Beans) — это спецификация, существенно расширившая возможности первоначальной компонентной модели Java Beans и разработанная в марте 1998 г. корпорацией Sun Microsystems. Если первоначальная модель была предназначена прежде всего для построения компонентов на стороне клиента, то EJB представляет собой фундамент для реализации ди-

динамически подключаемых серверных компонентов, которые позволяют расширять функциональность сервера во время выполнения. EJB обеспечивает более высокий уровень взаимодействия — уровень связи между удаленными готовыми компонентами, который, в свою очередь опирается на взаимодействие удаленных объектов клиента и сервера. Для Java-объектов это взаимодействие обеспечивает механизм удаленного вызова методов Remote Method Invocation (RMI), который во многом сходен с CORBA. Сервер объекта скрывает от клиента подробности его реализации, предоставляя доступ к объекту посредством интерфейса, написанного на языке Java.

В спецификации EJB заложена совместимость с CORBA. С другой стороны, в новой CORBA 3.0 присутствует спецификация компонентной модели CORBA Component Model — переход на более высокий уровень построения компонентных объектно-ориентированных приложений непосредственно с помощью CORBA. Причем эта компонентная модель обеспечивает отображение компонентов CORBA в Java Beans. Рассказывая о EJB, нельзя не сказать пару слов о языке программирования Java.

Язык Java произошел от языка программирования Oak, а не от C++, как думают многие. Согласно публикациям корпорации Sun Microsystems, выступающей в качестве разработчика языка Java, Oak был приспособлен для работы в Internet и затем переименован в Java.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Так же, как и библиотеки классов систем разработки приложений на языке C++, библиотеки классов Java значительно упрощают разработку приложений, предоставляя в распоряжение программиста мощные средства решения распространенных задач. Поэтому программист может больше внимания уделить решению прикладных задач, а не таких, как, например, организация динамических массивов, взаимодействие с операционной системой или реализация элементов пользовательского интерфейса.

Так вот ColdFusion в свою очередь позволяет вызывать распределенные объекты как на основе технологий COM, CORBA, EJB, так и объекты Java, при помощи тега <CFOBJECT>, описание которого мы рассмотрим ниже.

Тег <CFOBJECT>

Синтаксис тега <CFOBJECT> зависит от типа, определяемого с помощью атрибута TYPE, который может принимать следующие значения:

☐ COM

☐ CORBA

☐ JAVA

Судя по названию типа нетрудно догадаться, вызов объекта на основе какой технологии будет при этом использован.

Далее приведем синтаксис для каждого типа тега <CFOBJECT> и затем в табл. 21.1 объединим описание возможных атрибутов.

Синтаксис с вызовом объекта COM:

```
<CFOBJECT TYPE = "COM"
    NAME = "object name"
    ACTION = "Create | Connect"
    CLASS = "program_ID"
    CONTEXT = "InProc | Local | Remote"
    SERVER = "server_name">
```

Синтаксис с вызовом объекта CORBA:

```
<CFOBJECT TYPE = "CORBA"
    NAME = "object name"
    CLASS = "file or naming service"
    CONTEXT = "IOR | NameService"
    LOCALE = "type-value arguments">
```

Синтаксис с вызовом объекта EJB или Java:

```
<CFOBJECT TYPE = "Java"
    NAME = "object name"
    ACTION = "Create"
    CLASS = "Java class">
```

Таблица 21.1. Описание атрибутов тега <CFOBJECT>

Атрибут	Описание
ACTION	Действие. Обязательный атрибут для COM и Java и абсолютно неприменим для CORBA. Возможные значения для COM: • Create — создание объекта; данное значение также используется для EJB или Java; • Connect — соединение с объектом. Например: <pre><CFTRY> <CFOBJECT TYPE="COM" ACTION="Connect" CLASS="Word.Application" NAME="objWord"> <CFCATCH> <CFOBJECT TYPE="COM" ACTION="Create" CLASS="Word.Application"</pre>

Таблица 21.1 (продолжение)

Атрибут	Описание
	<pre> NAME="objWord"> </CFCATCH> </CFTRY> </pre> В представленном фрагменте кода если происходит ошибка на момент соединения с объектом, то, следовательно, данный объект не существует и его необходимо создать. Это, собственно говоря, мы и реализуем в блоке обработки ошибок
CLASS	Класс. Обязательный атрибут для всех типов. Для COM подразумевает использование уникального кода или наименования компонента подключаемого объекта. Для CORBA указывается в зависимости от контекста, определяемого атрибутом CONTEXT. Например: <pre> <CFOBJECT TYPE="CORBA" NAME="GetName" CONTEXT="IOR" CLASS="c:\\myobject.ior"> </pre> Для EJB или Java имя атрибута говорит само за себя — с помощью данного атрибута определяется класс Java
NAME	Имя, по которому в дальнейшем предполагается обращение к созданному объекту. Обязательный атрибут для всех типов
CONTEXT	Контекст, определяет характер доступа. Необязательный атрибут для COM и необходимый для CORBA. Для COM может принимать следующие значения: • InProc — смешанный; • Local — локальный; • Remote — удаленный. Для CORBA может принимать следующие значения: • IOR — используется Interoperable Object Reference (интероперабельная объектная ссылка) для доступа к CORBA-серверу; • NameService — используется наименование сервиса для доступа к серверу
SERVER	Сервер. Необходимо указывать в сочетании с контекстом только для COM. Может принимать одну из следующих форм: • www.allaire.com • http://www.allaire.com • 127.0.0.1 • ruben • \\ruben

Таблица 21.1 (окончание)

Атрибут	Описание
LOCALE	Устанавливает аргументы для обращения к <code>init_orb()</code>. Необязательный атрибут и если используется, то только для CORBA. Указание этого атрибута определяет VisiBroker orbs* и в настоящее время доступно на C++ версии 3.2. При этом атрибут должен иметь следующий вид: <pre>"-ORBagentAddr 199.99.129.33 -ORBagentPort 19000"</pre> где каждая пара значений должна начинаться с дефиса (-)

* VisiBroker orbs — это программный продукт, обеспечивающий полную поддержку спецификации CORBA.

Примеры вызова распределенных объектов

Что касается выбора распределенных объектов COM, CORBA или EJB, то автор не ставит перед собой цель склонить читателя к той или иной технологии. Например, для проектов средней сложности в среде Windows конечно следует использовать COM-технологию, тем более что в упомянутой операционной системе для этого все уже готово. Для сложных же масштабируемых проектов или при использовании гетерогенных сред, возможно, следует применять CORBA-технологию. Впрочем, и о EJB не стоит забывать. Однако в силу простоты демонстрации и привычки, выработанной годами, мы рассмотрим примеры вызова объектов Java и объектов, построенных по технологии COM.

Java-объекты

Прежде чем вызвать объект Java, его необходимо создать. Предположим, что требуется создать некий Java-объект, позволяющий производить определенные математические действия над двумя передаваемыми величинами. Для решения данной задачи формируем Java-код `Calculation.java`, представленный в листинге 21.1.

Листинг 21.1. Java-код `Calculation.java`

```
public class Calculation {
    public int x, y;
    public Calculation() {
        this.y = 1;
    }
}
```

```

    this.x = 1;
}

public int Sum(int x, int y) {return x+y;}
public int Subtraction(int x, int y) {return x-y;}
public int Multiplication(int x, int y) {return x*y;}
public int Division(int x, int y) {return x/y;}
}

```

где *x* и *y* — принимаемые в качестве параметров величины. Здесь также объявлены четыре метода, соответствующие каждому из математических действий: *Sum* (суммирование), *Subtraction* (вычитание), *Multiplication* (умножение) и *Division* (деление). Для данного кода было бы уместно включить определенные проверки (например, деление на ноль). Однако, дорогой читатель, все в ваших руках. А нам, между тем, необходимо продемонстрировать вызов данного объекта из ColdFusion.

Но перед тем как мы вызовем данный объект, надо бы совершить еще одно действие, а именно откомпилировать исходный текст Java из файла *Calculation.java* в байт-код Java-файла *Calculation.class*. Для подобных целей в Java существует компилятор *javac.exe*. В нашем случае следует выполнить команду:

```
Javac D:\CFUSION\Java\classes\Calculation.java
```

Данная команда предполагает, что наш класс размещается в каталоге *D:\CFUSION\Java\classes*, и для удобства использования новых классов этот каталог мы устанавливаем как месторасположение динамически загружаемых Java-классов (*Dynamic Class Load Path*), что позволяет не перезагружать службу ColdFusion для отображения очередных изменений, произведенных по отношению к выполняемому Java-классу.

Более подробно о настройках Java с использованием страницы администратора "ColdFusion Administrator" см. главу 14.

И теперь, когда подготовительная работа проведена, остается создать CFML-страницу, позволяющую вызвать объект Java и использовать существующие методы по назначению (листинг 21.2).

Листинг 21.2. Код страницы *MyCFOBJECTJava.cfm*

```

<HTML>
<HEAD>
    <TITLE>CFOBJECT Java</TITLE>
</HEAD>
<BODY>
    <CFOBJECT TYPE="Java"

```

```
ACTION="Create"  
CLASS="Calculation"  
NAME="myCalc">
```

```
<CFSET myCalc.x = 10>  
<CFSET myCalc.y = 5>  
<CFSET x = myCalc.x>  
<CFSET y = myCalc.y>  
<CFSET ResultSum = myCalc.Sum(x, y)>  
<CFSET ResultSubtraction = myCalc.Subtraction(x, y)>  
<CFSET ResultMultiplication = myCalc.Multiplication(x, y)>  
<CFSET ResultDivision = myCalc.Division(x, y)>  
<CFSCRIPT>  
    WriteOutput('x = ' & x & '; y = ' & y);  
    WriteOutput('<br>Sum = ' & ResultSum);  
    WriteOutput('<br>Subtraction = ' & ResultSubtraction);  
    WriteOutput('<br>Multiplication = ' & ResultMultiplication);  
    WriteOutput('<br>Division = ' & ResultDivision);  
</CFSCRIPT>  
</BODY>  
</HTML>
```

В результате выполнения представленной страницы будет выведено:

```
x = 10; y = 5  
Sum = 15  
Subtraction = 5  
Multiplication = 50  
Division = 2
```

К этому, собственно говоря, мы и стремились.

COM-объекты

Смысл использования объектов, построенных по технологии COM, заключается в следующем. В операционной системе зарегистрированы программы, которые могут выступать в роли COM-объектов, предоставляющих другим программам возможности использования своих объектов. ColdFusion может выступать в роли одной из таких программ, использующих COM-объекты других приложений. Впрочем, вы также можете создавать собственные COM-объекты, но мы в нашем примере будем вызывать уже существующие распределенные объекты, тем более что особого дефицита в COM-

объектах в среде Windows нет. Если учесть, что только от Microsoft их несколько десятков.

Предположим, что нам необходимо поместить данные из существующей базы данных *Cruise* в файл Microsoft Excel, используя при этом доступ к базе данных с помощью объекта ADO DB (ActiveX Data Objects DataBase). Отметим, что ADO DB представляет собой объект, являющийся неотъемлемой частью библиотеки типов ADO (ActiveX Data Objects) от Microsoft, и применяется для разработки соединений баз данных, выполнения команд и извлечения наборов данных из источников.

Как уже было отмечено в описании тега <CFOBJECT>, в качестве атрибута CLASS при использовании TYPE="COM" необходимо указывать уникальный код или название компонента подключаемого объекта. И прежде чем формировать CFML-страницу с вызовом распределенного объекта, необходимо выяснить этот самый уникальный код или имя объекта.

В общем, здесь есть несколько путей, один из них — обратиться непосредственно к редактору реестра (Regedit.exe) и в разделе HKEY_CLASSES_ROOT\CLSID\, просмотрев список COM-объектов, найти необходимый код объекта. Однако, если у вас установлен пакет Microsoft Visual Studio версии 6.0, то просмотр зарегистрированных COM-объектов легко решается с помощью утилиты OLE/COM Object Viewer (Oleview.exe).

Перед тем как продемонстрировать работу с инструментом просмотра OLE- и COM-объектов, хотелось бы внести некоторое пояснение к работе с базами данных при использовании ADO DB. При подключении существующей базы данных необходимо применить метод Open коллекции Connection объекта ADO DB, после чего следует создать ссылку на набор данных с использованием метода Open только уже коллекции Recordset все того же объекта ADO DB.

На примере коллекции Connection объекта ADO DB найдем необходимый для вызова объекта — уникальный код и научимся определять синтаксис методов, где важно правильно задать значения, передаваемые в качестве параметров.

Для работы с программой просмотра OLE- и COM-объектов в качестве первого шага необходимо запустить программу Oleview.exe, расположенную в папке X:\Program File\Microsoft Visual Studio\Common\Tools\, где X является устройством, куда была установлена программа MS Visual Studio 6.0. После запуска OLE/COM Object Viewer выберем пункт **All Objects** (Все объекты), как показано на рис. 21.1.

Следующим шагом находим сам объект ADODB.Connection (рис. 21.2).

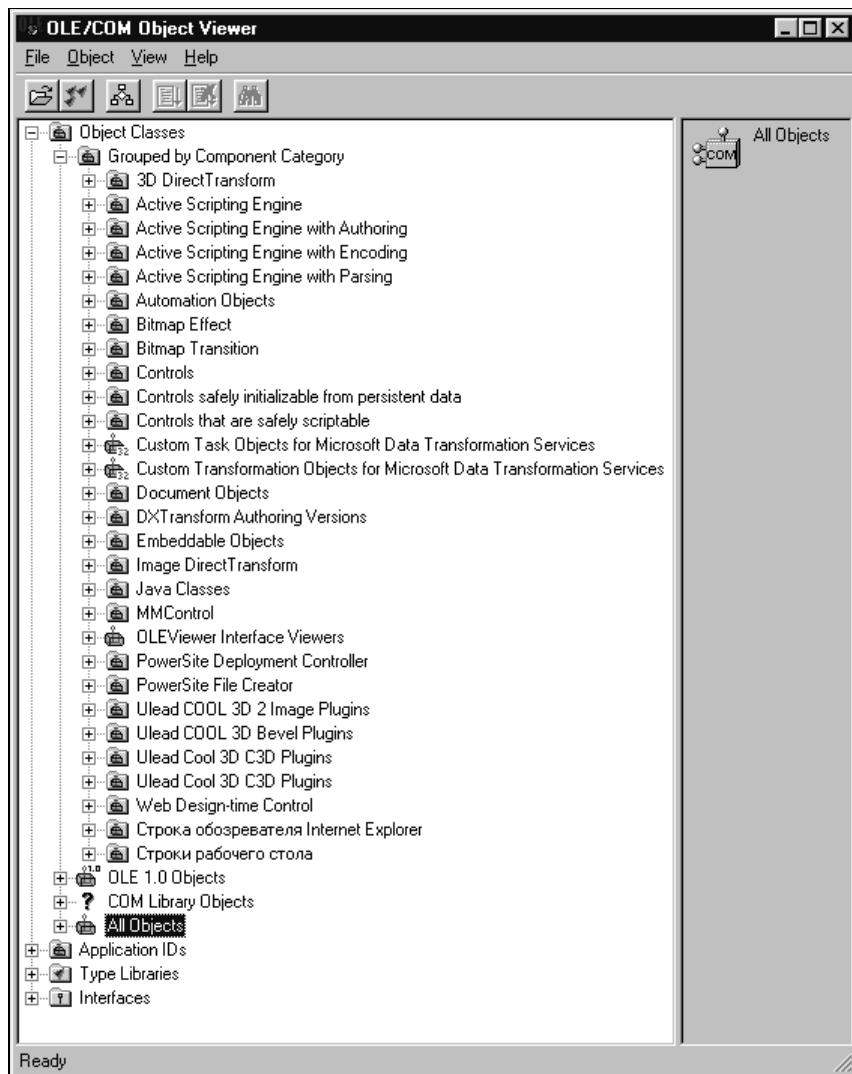


Рис. 21.1. OLE/COM Object Viewer, пункт **All Objects**

Как видно из рис. 21.2, здесь же указан тот самый уникальный код объекта, который нет необходимости запоминать, достаточно щелкнуть правой кнопкой мыши по имени объекта и из контекстного меню выбрать команду **Copy CLSID to Clipboard** (Скопировать код в буфер обмена). В нашем случае в буфере обмена будет помещен следующий фрагмент:

```
{00000514-0000-0010-8000-00AA006D2EA4}
```

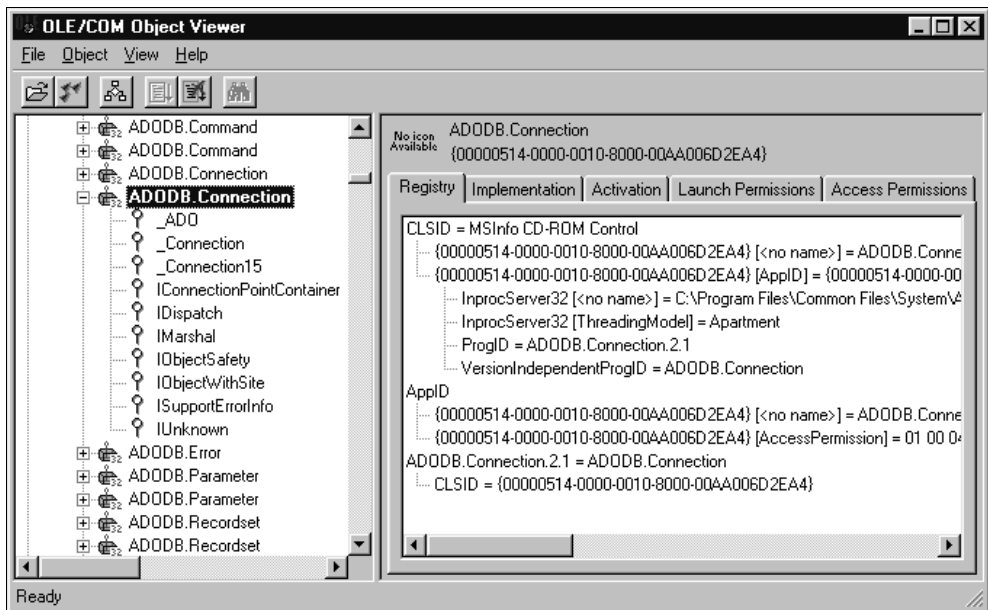


Рис. 21.2. OLE/COM Object Viewer, пункт **All Objects**, объект **ADODB.Connection**

Теперь можно смело создать код, позволяющий вызывать объект ADO DB с коллекцией Connection:

```
<CFOBJECT TYPE="COM"
ACTION="Create"
CLASS="{00000514-0000-0010-8000-00AA006D2EA4}"
NAME="objConn">
```

Конечно, можно было бы написать иначе, например:

```
<CFOBJECT TYPE="COM"
ACTION="Create"
CLASS="ADODB.Connection"
NAME="objConn">
```

В последнем фрагменте выполнено обращение к объекту по имени **ADODB.Connection**, но заметьте, что на рис. 21.2 объектов с подобным именем два, и при этом каждый имеет свой уникальный номер и использует собственную версию библиотеки (**msado10.dll** и **msado15.dll**). Где гарантия, что в последнем случае использования тега <CFOBJECT> будет вызван именно тот COM-объект, который вам необходим? То же относится и к объекту **ADODB.Recordset**. А вот объект офисного приложения MS Excel можно смело вызывать по имени — **Excel.Application**.

Например:

```
<CFOBJECT TYPE="COM"
    ACTION="Create"
    CLASS="Excel.Application"
    NAME="objExcel">
```

Если вы прочитали предыдущую главу, то знаете, что в CFML-страницах можно использовать CFScript, и нам ничто не мешает вызвать объект Excel.Application с помощью функции CreateObject внутри блока <CFSCRIPT>, как показано в следующем примере:

```
<CFSCRIPT>
    objExcel = CreateObject("COM", "Excel.Application");
</CFSCRIPT>
```

Впрочем, вернемся к инструменту просмотра OLE- и COM-объектов. Дело в том, что мало вызвать объект, по отношению к нему надо совершать какие-либо действия. И для просмотра этих самых действий в виде методов, надо выполнить следующее:

1. В контексте выбранного объекта выделить подраздел, например все для того же ADODB.Connection выделить IDispatch.
2. Щелкнув по выбранному подразделу правой кнопкой мыши, выбрать команду **View** (Просмотр).
3. В открывшемся диалоговом окне нажать кнопку **View TypeInfo** (Просмотр информации о типе).
4. В появившемся окне **ITypeInfo Viewer** (Просмотр информации о типе) раскрыть список методов (**Methods**) и выбрать необходимый для изучения метод, как показано на рис. 21.3.

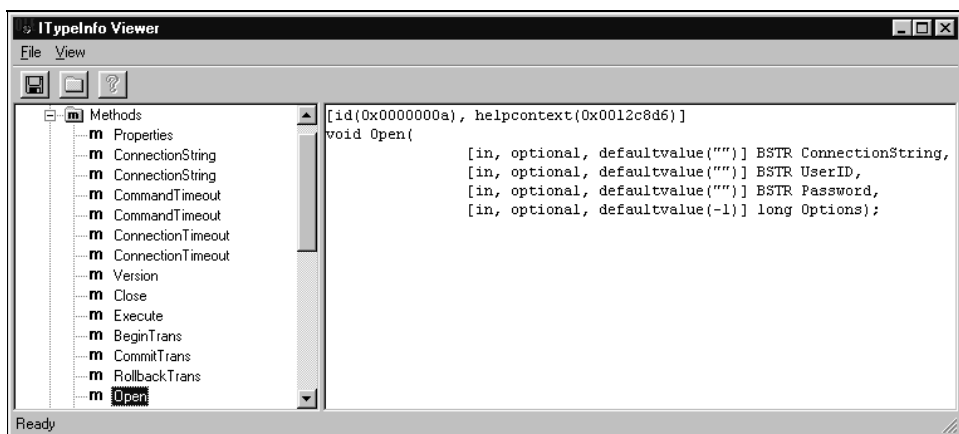


Рис. 21.3. ITypeInfo Viewer, пункт **Methods**, метод Open

Как видно из рис. 21.3, методу `Open` объекта `ADODB.Connection` в качестве передаваемых параметров необходимо назначить:

- ☐ `ConnectionString` — имя ODBC-источника, для которого будет происходить соединение;
- ☐ `UserID` — идентификатор пользователя;
- ☐ `Password` — пароль пользователя;
- ☐ `Options` — режим соединения, принимающий по умолчанию значение `-1`.

Например по Internet-адресу (рис. 21.4) <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdcstlocktypeenum.asp> можно узнать, что режим соединения способен принимать два значения: `-1` (`synchronously` (синхронный)) и `16` (`asynchronously` (асинхронный)).

The screenshot shows the Microsoft Internet Explorer browser window with the address bar displaying <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdcstlocktypeenum.asp>. The page content includes the MSDN logo, navigation links, and a search bar. The main content area displays the 'ADO Enumerated Constants' page, which lists various constants and provides a detailed description of the 'ConnectOptionEnum' constant.

ADO Enumerated Constants - Microsoft Internet Explorer

Address: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdcstlocktypeenum.asp>

msdn Microsoft

MSDN Home | MSDN Library | Downloads | Code Center | Site Map | MSDN Worldwide

Search Advanced Search

All MSDN Library Only

sync toc

Up One Level

- ☐ ADCPROP_ASYNCTHREADI
- ☐ ADCPROP_AUTORECALC_E
- ☐ ADCPROP_UPDATECRITER
- ☐ ADCPROP_UPDATERESYN
- ☐ AffectEnum
- ☐ BookmarkEnum
- ☐ CommandTypeEnum
- ☐ CompareEnum
- ☐ ConnectModeEnum
- ☐ ConnectOptionEnum
- ☐ ConnectPromptEnum
- ☐ CopyRecordOptionsEnum
- ☐ CursorLocationEnum
- ☐ CursorOptionEnum
- ☐ CursorTypeEnum
- ☐ DataTypeEnum
- ☐ EditModeEnum
- ☐ ErrorValueEnum
- ☐ EventReasonEnum
- ☐ EventStatusEnum
- ☐ ExecuteOptionEnum
- ☐ FieldEnum

MSDN Home > MSDN Library > Microsoft ActiveX Data Objects (ADO) > ADO Programmer's Reference > ADO API Reference > ADO Enumerated Constants

ADO 2.7 API Reference

ConnectOptionEnum

Specifies whether the `Open` method of a `Connection` object should return after (`synchronously`) or before (`asynchronously`) the connection is established.

Constant	Value	Description
adAsyncConnect	16	Opens the connection <code>asynchronously</code> . The <code>ConnectComplete</code> event may be used to determine when the connection is available.
adConnectUnspecified	-1	Default. Opens the connection synchronously.

ADO/WFC Equivalent

Package: **com.ms.wfc.data**

Constant
<code>AdoEnums.ConnectOption.ASYNCCONNECT</code>
<code>AdoEnums.ConnectOption.CONNECTUNSPECIFIED</code>

See Also

Applies To: `Open` Method (ADO Connection)

©1998-2001 Microsoft Corporation. All rights reserved.

ADO Enumerated Constants

Internet

Рис. 21.4. Описание методов COM-объектов в Internet

Очередной пример демонстрирует вызов объекта `ADODB.Connection` с последующим открытием соединения в синхронном режиме:

```
<CFOBJECT TYPE="COM"
    ACTION="Create"
    CLASS = "{00000514-0000-0010-8000-00AA006D2EA4}"
    NAME="objConn">

<CFSCRIPT>
    objConn.Open("Cruise", "dba", "sql", -1);
</CFSCRIPT>
```

Если вы еще не забыли, что изначально перед нами стояла задача поместить данные из существующей базы данных `Cruise` в файл `Microsoft Excel`, осуществляя при этом доступ к базе данных с помощью объекта `ADO DB`, то самое время об этом вспомнить. И теперь, когда вы имеете некоторое представление об использовании `COM`-объектов, представим листинг решения описанной задачи.

Листинг 21.3. Код страницы `MyCFOBJECTCOM.cfm`

```
<HTML>
<HEAD>
    <TITLE>CFOBJECT COM</TITLE>
</HEAD>
<BODY>
    <CFOBJECT TYPE="COM"
        ACTION="Create"
        CLASS="Excel.Application"
        NAME="objExcel">

    <!-- ADODB.Connection -->
    <CFOBJECT TYPE="COM"
        ACTION="Create"
        CLASS = "{00000514-0000-0010-8000-00AA006D2EA4}"
        NAME="objConn">

    <!-- ADODB.Recordset -->
    <CFOBJECT TYPE="COM"
        ACTION="CREATE"
        CLASS="{00000535-0000-0010-8000-00AA006D2EA4}"
        NAME="objRecord">
```

```
<CFSCRIPT>
// Определение наименования и месторасположения файла
File = "Cruise.xls";
FilePath = GetDirectoryFromPath(GetCurrentTemplatePath()) & File;
if (FileExists(FilePath) EQ "true") {
    WriteOutput("File: '" & FilePath & "' exist!");
} else {
    pmCursorType = 3;
    pmLockType = 1;
    pmOptions = 1;
    pmRows = -1;
// Открытие соединения ADO DB и выполнение запроса
    objConn.Open("Cruise", "dba", "sql", -1);
    strSQL = "SELECT Name_tou, PointDepart_tou FROM Tour";
    objRecord.Open(strSQL, objConn, pmCursorType,
        pmLockType, pmOptions);
// Создание объекта Excel
    objWorkBook = objExcel.Workbooks;
    objOpenedBook = objWorkBook.Add();
    objWorkSheets = objExcel.WorkSheets;
    objWorkSheet = objWorkSheets.Add();
// Получение данных
    arRows = objRecord.GetRows(pmRows);
// Формирование шапки таблицы в Excel
    objRange = objExcel.Range("A1");
    objRange.ColumnWidth = 25;
    objRange = objExcel.Range("B1");
    objRange.ColumnWidth = 30;
    objRange = objExcel.Range("A1:A1");
    objRange.Value = "Name Tour";
    objRange = objExcel.Range("B1:B1");
    objRange.Value = "Point Depart";
// Цикл заполнения данными в Excel
    iCurRow = 1;
    while (iCurRow LE objRecord.RecordCount) {
        for (iCurCol = 1; iCurCol LE 2; iCurCol = iCurCol+1) {
            iCurRowExl = iCurRow + 1;
            iCurColExl = Iif(iCurCol EQ 1,DE("A"),DE("B"));
```

```

        objRange =
objExcel.Range("#iCurColExl##iCurRowExl#:#iCurColExl##iCurRowExl#");
        objRange.Value = arRows[iCurCol][iCurRow];
    }
    iCurRow = iCurRow + 1;
}

// Сохранение файла Excel и закрытие созданных объектов
objWorkSheet.SaveAs(FilePath, Val(1));
objWorkBook.Close();
objExcel.Quit();
objExcel = "Nothing";
objRecord.Close();
objConn.Close();
}
</CFSCRIPT>
<META HTTP-EQUIV="Refresh"
        CONTENT="0;URL=<CFOUTPUT>#File#</CFOUTPUT>">
</BODY>
</HTML>

```

На рис. 21.5 изображен результат выполнения приведенного выше кода.

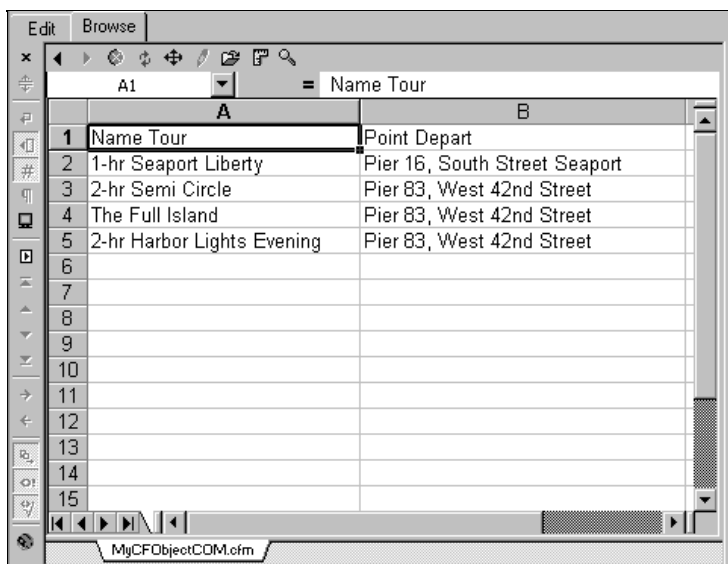


Рис. 21.5. Результат выполнения страницы MyCFOBJECTCOM.cfm

В качестве самостоятельной работы можете изучить такие COM-объекты, как:

- ☐ Allaire CfmlValidator Object;
- ☐ Allaire CFSRdsServerInfo Class;
- ☐ Allaire Validator Object;
- ☐ Allaire VeritySearch Class;
- ☐ Macromedia Flash Factory Object;
- ☐ Macromedia FTP & RDS.

Резюме

Подведем итог. В данной главе мы использовали CFML-тег <CFOBJECT>.



Использование WML-страниц для разработки WAP-приложений

Если вы разработчик Web-сайтов, не исключено что в силу бурного развития мобильной связи в скором времени вам доведется задуматься над проектированием WAP-сайта. Но обо всем по порядку.

Если вам доводилось когда-либо просматривать каталоги современных мобильных телефонов, то вы наверняка обращали внимание на следующие фразы: "поддержка WAP" или "встроенный микробраузер с поддержкой технологии WAP". Эта фраза предполагает, что данный телефон позволит вам производить заказы на различные услуги, бронировать билеты или выполнять определенные операции, используя протокол беспроводных приложений — WAP (Wireless Application Protocol). Впрочем, у автора есть подозрение, что каждый второй читатель уже обладает подобным мобильным устройством.

За последнее время мы стали свидетелями возникновения и бурного развития мобильной связи. Сегодня многие корпорации провозглашают приоритетные направления в развитии и совершенствовании собственных технологий на базе информационной интеграции и технологической координации за счет существенного расширения специализированного информационно-организационного сервиса в Internet. При этом совершенно уникальные возможности для обеспечения истинной мобильности в управлении бизнесом открываются благодаря появлению протокола беспроводных приложений WAP и WAP-Internet-технологии.

Первой компанией, предложившей решения для беспроводного доступа к данным в Internet, явилась Unwired Planet, преобразовавшаяся в дальнейшем в компанию Phone.com. Уже в конце 1997 г. был организован WAP Forum, в состав которого вошли все ведущие производители беспроводных устройств, коммуникационного оборудования, операторы беспроводной связи и всех стандартов, а также производители программного и аппаратного обеспечения. WAP Forum, в свою очередь, разработал открытую WAP-спецификацию, предназначенную для стандартизации беспроводных протоколов, с помощью которых мобильные устройства получают доступ к различным услугам сети. В 1999 г. была принята наиболее распространенная версия

спецификации WAP 1.1 и уже к третьему кварталу 2000 г. была принята спецификация версии 1.2, являющаяся дополнением к предыдущей версии.

Появление WAP расширяет возможности Internet, добавляя новую грань — мобильность. Уже сегодня существует целый ряд транспортных услуг с использованием WAP-технологий. Например, в Лондоне развит мобильный информационный участок, который собирает воедино информацию об услугах лондонского общественного транспорта для обеспечения легкого доступа к ней посредством мобильных телефонов. С помощью этой службы вы можете:

- ☐ узнать последние новости метро, включая задержки поездов и закрытия станций;
- ☐ получить сводку за более длительный период о лондонских автобусах и речном транспорте;
- ☐ получить доступ к расписанию движения поездов железной дороги и ознакомиться с информацией о происшествиях;
- ☐ найти информацию о телефонах горячей линии, потерянных вещах и другую полезную информацию для пассажиров Великобритании.

Однако продолжим. Имея общее представление о концепции сети Internet, нетрудно разобраться в технологии WAP, построенной по аналогии. В WAP, также как и Web, существуют понятия браузера, протокола, шлюза, сервера и языка описания передаваемых данных.

WAP-браузер встроен в беспроводные устройства (вспомните каталоги современных мобильных телефонов). Впрочем, на сегодняшний день существует достаточное количество так называемых эмуляторов WAP-браузера в виде программного обеспечения для компьютеров, в некоторой литературе их еще называют имитаторами WAP-браузера, но смысл при этом не меняется.

WAP-протокол использует технологию, очень схожую с применяемой для передачи данных по компьютерным сетям и в частности по Internet.

С точки зрения разработчика программного обеспечения клиентом является не само мобильное устройство, а программный компонент, называемый WAP-шлюзом, который преобразует запросы мобильного устройства в запросы HTTP и перенаправляет HTTP-ответы мобильному устройству через все тот же WAP-шлюз.

WAP-сервер — это, по сути, все тот же Web-сервер, требующий дополнительной конфигурации.

В силу ограничений на форматы и на объемы передаваемых данных для WAP в качестве языка описания передаваемых данных был разработан WML (Wireless Markup Language) — "диалект" HTML/XML. Аналогом же языка описания сценариев JavaScript для WAP является WMLScript, который по-

звolyет программировать логику и получать доступ к некоторым функциям мобильных устройств. Дальше, больше!

Подготовка к работе

В первую очередь нам необходимо настроить HTTP-протокол для совместной работы с WAP-шлюзом и мобильными устройствами, поддерживающими технологию WAP. (Мы опишем подобную настройку для двух наиболее популярных серверов: Apache и Microsoft IIS.) Затем для удобства отладочного процесса необходимо установить один из эмуляторов WAP-браузера, о которых будет сказано несколько слов.

Конфигурация типов MIME для Web-сервера

Для конфигурации MIME-типов сервера Apache необходимо:

1. Открыть файл `mime.types`. Полный путь месторасположения данного файла может быть следующим: `C:\Apache\conf`.
2. Внести изменения в формате "MIME-тип/расширение", как показано ниже:

# MIME type	Extension
<code>text/vnd.wap.wml</code>	<code>wml</code>
<code>image/vnd.wap.wbmp</code>	<code>wbmp</code>
<code>application/vnd.wap.wmlc</code>	<code>wmlc</code>
<code>text/vnd.wap.wmlscript</code>	<code>wmls</code>
<code>application/vnd.wap.wmlscriptc</code>	<code>wmlsc</code>

3. Сохранить этот файл и перезапустить сервер Apache.

Для конфигурации MIME-типов сервера Microsoft IIS необходимо:

1. Запустить редактор реестра `Regedit.exe`.
2. Открыть ключи реестра:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\MimeMap
```

3. Добавить новые значения пар, описанные ниже (используйте для этого команду **Edit | New | String Value** (Правка | Создать | Строковый параметр)):

Value name	Value data
<code>"text/vnd.wap.wml, wml"</code>	<code>".wml"</code>
<code>"image/vnd.wap.wbmp, wbmp"</code>	<code>".wbmp"</code>
<code>"application/vnd.wap.wmlc, wmlc"</code>	<code>".wmlc"</code>
<code>"text/vnd.wap.wmlscript, wmls"</code>	<code>".wmls"</code>
<code>"application/vnd.wap.wmlscriptc, wmlsc"</code>	<code>".wmlsc"</code>

4. После чего перезагрузить сервер Microsoft IIS.

Эмуляторы WAP-браузера

Пока WAP-страницы нельзя просматривать традиционными Web-браузерами, но может быть, недалек тот час, когда очередные версии Microsoft Internet Explorer и Netscape Navigator предоставят такую возможность, а пока следует использовать специализированные программы.

Во-первых, эмуляторы WAP-браузера поставляются вместе со специализированными пакетами разработок WAP-сайтов, такими как:

☐ Nokia WAP Toolkit; ☐ Ericsson WapIDE и т. д.

Во-вторых, существуют отдельные эмуляторы WAP-браузера, например:

☐ Wireless Companion; ☐ M3Gate и т. д.

В-третьих, нельзя забывать Internet, где на обычном Web-сервере, например таком как по адресу <http://www.gelon.net>, можно выбрать любую доступную модель WAP-телефона и в открывшемся окне, представленном в виде образа телефона, нажимать кнопки этого телефона, используя компьютерную мышь, а с помощью клавиатуры вводить адреса WAP-сайтов, которые будут высвечиваться на дисплее виртуального сотового аппарата.

В связи с тем, что пакет ColdFusion Studio поддерживает язык разметки для беспроводных устройств — Wireless Markup Language (WML) и является самодостаточным в плане разработки WML-страниц, нам остается только установить один из эмуляторов WAP-браузера, чтобы просматривать результат выполнения WML-страниц на локальном сервере. Например, воспользуемся WAP-браузером Wireless Companion от компании YourWAP.com. На первом этапе этого будет достаточно.

Обзор языка WML

Как уже было отмечено, для WAP в качестве языка описания передаваемых данных был разработан WML (Wireless Markup Language), "диалект" HTML/XML. По этой причине для разработчиков Web-сайтов не составит особой сложности изучить основы языка WML.

Суть разработки WML-страницы сводится к формированию некоей колоды карт, где пользователь одновременно видит содержание одной карты (card), но при этом он может обращаться к другим картам из той же колоды (deck), не требуя взаимодействия с сервером, т. к. вместе с активной картой в память устройства загружается вся колода в виде WML-файла. Переход же по картам осуществляется с использованием определенного синтаксиса: #NameCard. Впрочем, пользователь может обращаться и к картам из других колод: NameFile.wml#NameCard, но на это может уйти некоторое время, поскольку в этом случае устройство попытается обратиться к серверу для загрузки очередной колоды — WML-файла.

Примечание

Размер загружаемых в WAP-устройства файлов ограничен объемом порядка 1,4 Кбайт, максимальные границы могут варьироваться в зависимости от модели устройства.

В ColdFusion Studio существует возможность создать новый документ, используя шаблон документа WML. Для этого достаточно выбрать команду **File | New** (Файл | Создать) и в открывшемся диалоговом окне **New Document** (Новый документ) на вкладке **WML** выбрать шаблон **Static** (Статический). После чего в основном окне редактирования откроется файл со следующим содержанием:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd">
<wml>
  <card>

  </card>
</wml>
```

Таким образом, у нас уже есть колода (deck) с одной картой (card) и необходимый для составления WML-страниц пролог. Пролог включает в себя ссылку на файл DTD (Data Type Definition), позволяющую определить структуру и состав используемого языка. Колода, как вы наверно заметили, задается парой тегов `<wml>` и `</wml>`, карта колоды, соответственно находящаяся внутри колоды, определяется парой тегов `<card>` и `</card>`. Внутри же самой карты вы можете определять ее содержание. Например:

```
<card>
  <p>Text</p>
</card>
```

Заметим, что для форматирования текста могут использоваться теги, аналогичные HTML. Теги `<em>` и `<i>` позволяют выделять текст курсивом. Тег `<u>` предназначен для подчеркивания текста. Теги `<b>` и `<strong>` используются для выделения текста жирным шрифтом. Теги `<big>` и `<small>` позволяют увеличивать и уменьшать размер текста. Тег `<br/>`, не имеющий пары, предназначен для разрыва строки. Тег же `<table>` позволяет выстраивать таблицы с уже привычными `<tr>` и `<td>`.

Использованный нами тег `<p>` определяет параграф. Более того, для параграфа можно задавать атрибут выравнивания текста `align`, принимающий значения: `center` (по центру), `left` (по левому краю) или `right` (по правому краю). Также для параграфа можно задавать способ переноса текста атрибутом `mode`, принимающим значения `wrap` (с автопереносом) или `nowrap`

(без автопереноса). Обратите внимание на рис. 22.1, где показано окно редактирования документа в ColdFusion Studio с выводом меню автоматической простановки тегов для определения параграфа.

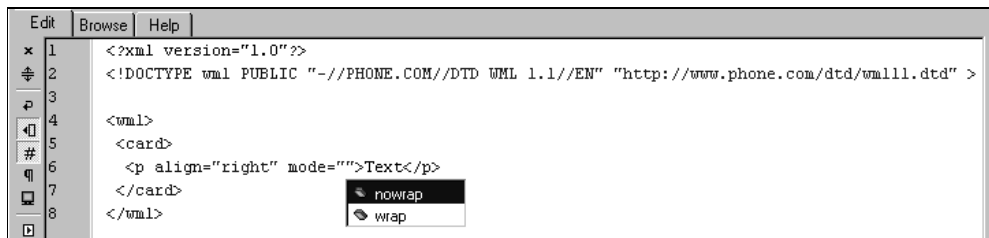


Рис. 22.1. Пример использования ColdFusion Studio в качестве редактора WML-страниц

Внимание

Использование больших букв в именах тегов и атрибутах этих тегов может приводить к возникновению ошибок.

В раздел колоды можно включать несколько карточек, при этом, вызывая WML-страницу, на экране устройства активизируется первая карта. Что бы осуществлять переход по карточкам, каждой карте необходимо указать идентификатор. Для этого существует атрибут `id`. Например:

```
<card id="Cruises" title="My-Line Cruises">
  <p mode="wrap">
    New York's <i>BEST</i><br/>
    Sightseeing and Entertainment Cruises
  </p>
</card>
```

В приведенном примере помимо идентификатора карты `id` также определен заголовок с помощью атрибута `title`, который позволяет отображать текст в верхней части экрана при активизации данной карты. И теперь, когда наша карта уже приняла допустимый вид, можно приступить к созданию других карт. Например, следующая колода представлена двумя картами `Cruises` и `Schedules`, с возможностью перемещаться по этим картам, благодаря знаменитому тегу `<a>`, позволяющему устанавливать ссылки как на карты внутри колоды, так и на карты других колод:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM/DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd" >
<wml>
  <card id="Cruises" title="My-Line Cruises">
```

```

<p mode="wrap">
  New York's <i>BEST</i><br/>
  Sightseeing and Entertainment Cruises
  <a href="#Schedules">Schedules</a><br/>
</p>
</card>

<card id="Schedules" title="Schedules">
  <p>
    <u>Full Island Cruise</u><br/>
    <i>Oct.5-Nov.1 (Mon.-Fri.)</i><br/>
    10:00 am<br/>
    11:00 am<br/>
    1:30 pm<br/>
    2:30 pm<br/>
    <a href="#Cruises">Cruises</a><br/>
  </p>
</card>
</wml>

```

Тег `<a>` имеет укороченный синтаксис от тега `<anchor>`. На рис. 22.2 и 22.3 отображены экранные формы устройства при загрузке каждой из карт, представленных в описанном примере.

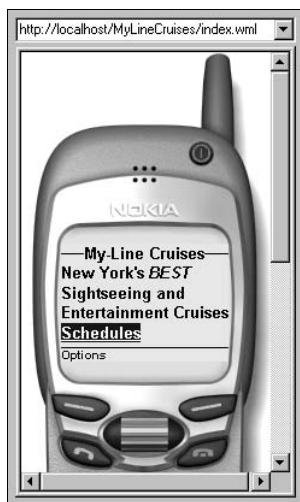


Рис. 22.2. Результат выполнения карты Cruises



Рис. 22.3. Результат выполнения карты Schedules

Далее поговорим об управлении ассоциативными программными кнопками с определенными командами. В мобильных устройствах достаточно часто используются левая и правая клавиши для выполнения тех или иных действий. Так вот, язык WML позволяет программировать данные действия. И здесь типовыми действиями являются:

- ❑ `accept` — добавляется собственная опция с соответствующим действием;
- ❑ `go` — безусловный переход к требуемой карте или колоде;
- ❑ `prev` — переход к предыдущей карте;
- ❑ `refresh` — обновление содержимого дисплея;
- ❑ `noop` — пустое действие.

Прежде чем продемонстрировать пример, отметим, что данные действия могут использоваться в качестве значения атрибута `type` тега `<do>`. Причем перечисленные действия могут быть определены как на уровне конкретной карты, так и на уровне всей колоды с использованием тега `<template>`. Например, несколько видоизменим предыдущий пример:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
  "http://www.phone.com/dtd/wml11.dtd" >

<wml>
  <template>
    <do type="accept" name="Author" label="Author" optional="false">
      <go href="#Author" sendreferer="false" method="get"/>
    </do>
    <do type="prev" name="BACK" label="Back"
      optional="false"><prev/></do>
  </template>

  <card id="Cruises" title="My-Line Cruises">
    <do type="accept" name="List" label="List Cruises" optional="false">
      <go href="ListCruises.wml#List" sendreferer="false" method="get"/>
    </do>
    <p mode="wrap">
      New York's <i>BEST</i><br/>
      Sightseeing and Entertainment Cruises
      <a href="#Schedules">Schedules</a><br/>
    </p>
  </card>

  <card id="Schedules" title="Schedules">
```

```

<p>
<u>Full Island Cruise</u><br/>
<i>Oct.5-Nov.1 (Mon.-Fri.)</i><br/>
10:00 am<br/>
11:00 am<br/>
1:30 pm<br/>
2:30 pm<br/>
<a href="#Cruises">Cruises</a><br/>
</p>
</card>

<card id="Author" title="Author">
<p>
<small>Ruben Akhayan</small><br/>
<a href="#Cruises">Cruises</a><br/>
<a href="#Schedules">Schedules</a><br/>
</p>
</card>
</wml>

```

Отметим, что в приведенном примере перед определением карт мы установили действие `prev`, позволяющее, находясь на любой карте текущей колоды, возвращаться к предыдущей карте. Также в области определения действий для всех карт `<template>` было определено действие перехода на карту `Author`, расположенную в этой же колоде. А вот уже непосредственно для карты `Cruises` было определено действие перехода на карту `List` другой колоды — `ListCruises` в виде загрузки файла `ListCruises.wml`. Сравните рис. 22.3 и 22.4, где на последнем в нижней части дисплея появились два дополнительных пункта — **Options** (Параметры) и **Back** (Назад), позволяющих совершать определенные действия.

На рис. 22.5 и 22.6 отображены экранные формы устройства при активизации пункта **Options** (Параметры) в зависимости от текущей карты `Cruises` и `Schedules`.

Иногда возникает необходимость выполнить определенные действия по прошествии заданного интервала времени, начиная с определенного момента. На такой случай в WML предусмотрен элемент таймера `timer`. Элемент таймер "следит" за временем и по истечении заданного интервала времени генерирует событие `ontimer`. Например, перед загрузкой основной колоды на протяжении трех секунд продемонстрируем пользователю наш логотип, сформировав страницу под названием `start.wml`.

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"

```

```
"http://www.phone.com/dtd/wml11.dtd" >
```

```
<wml>
  <card id="start" ontimer="index.wml">
    <timer value="3"/>
    <p align="center">
    <br/><i>One moment ...</i>
  </p>
</card>
</wml>
```

В нашем случае на дисплее мобильного устройства может быть выведен логотип, как показано на рис. 22.7.

Обращаем ваше внимание, что в языке WML, так же, как и в HTML, используется похожий тег `<img/>`, где в качестве атрибута `src` включается графический образ, с одной лишь разницей — данный образ имеет формат WBMP (Wireless BMP). Для конвертирования в формат WBML лучше всего выбирать небольшие изображения. Максимальный размер картинки зависит от устройства, но при всем при том не должен превышать 1,4 Кбайт. Впрочем, не следует злоупотреблять графическими образами из-за небольшого количества памяти и чрезмерно длительного ожидания загрузки файла.



Рис. 22.4. Результат выполнения карты **Schedules** с определением действий

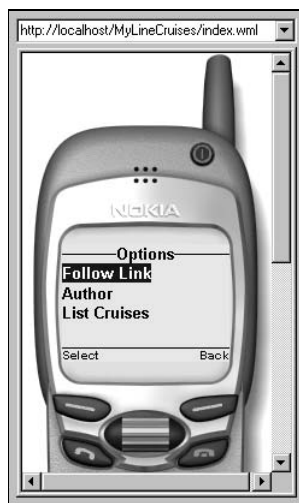


Рис. 22.5. Вывод списка опций **Options** для карты **Cruises**



Рис. 22.6. Вывод списка опций **Options** для карты **Schedules**



Рис. 22.7. Вывод логотипа на определенный промежуток времени

В WML так же, как и во всех языках, реализована концепция переменных. Например, в зависимости от значения той или иной переменной можно изменять и поведение карты. Значения переменным могут быть присвоены как посредством тега `<setvar>`, так и с использованием элементов ввода `<input>`, `<option>`, `<select>` и др. В следующем примере представлена форма для идентификации пользователя, где предварительно объявляются необходимые переменные:

```
<card id="BuyTickets" title="Buy Tickets">
  <onevent type="onenterforward">
    <refresh>
      <setvar name="EMail" value="Test@MyMail" />
      <setvar name="Password" value="Test" />
      <setvar name="Category" value="Adults" />
    </refresh>
  </onevent>
  <onevent type="onenterbackward">
    <refresh>
      <setvar name="EMail" value="" />
      <setvar name="Password" value="" />
      <setvar name="Category" value="" />
    </refresh>
  </onevent>
</card>
```

```
</onevent>
<do type="prev" label="Home"><go href="#Init"/></do>
<p align="left" mode="nowrap">
E-Mail: <input type="text" name="EMail"/><br/>
Password: <input type="password" name="Password"/><br/>
Category:
<select name="Category">
  <option value="Adults">Adults</option>
  <option value="Children">Children</option>
  <option value="Seniors">Seniors</option>
</select>
<anchor title="go">Submit<go method="post"
      href="http://localhost/MyLineCruises/login.cfm">
</go></anchor>
</p>
</card>
```

В данном примере представлено содержание одной карты BuyTickets, где пользователю предлагается ввести его электронный адрес, пароль и возрастную категорию. Как видно из примера, применение элементов ввода `<input>`, `<option>` и `<select>` схоже с использованием подобных элементов в HTML. В этом же примере было применено отслеживание событий для объявления переменных. Поскольку значения переменных не уничтожаются при переходе от одной карты к другой, их можно использовать в различных картах. Это обстоятельство может рассматриваться как преимущество, хотя не всегда. При переходе от карты к карте фактические значения элементов карты могут начать расходиться с ее реальным значением. Во избежание этого разработано несколько событий (тег `<onevent>`), позволяющих отслеживать так называемый "маршрут", по которому пользователь "дошел" к текущей карте. Для нашего примера тип `onenterforward` подразумевает событие, происходящее при переходе "вперед", а тип `onenterbackward` предполагает событие, происходящее при переходе "назад".

На рис. 22.8 показан результат выполнения примера по идентификации пользователя, а на рис. 22.9 продемонстрирован способ работы со списками.

Мы ни слова не сказали о WMLScript, только упомянув, что он является аналогом JavaScript. Впрочем, это уже отдельная тема. Скажем только, что WMLScript является процедурным языком и поддерживает локально установленные стандартные библиотеки.

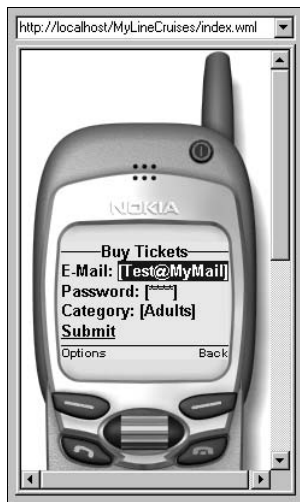


Рис. 22.8. Результат выполнения карты BuyTickets



Рис. 22.9. Пример работы со списками

Совместное использование WML и CFML

Для того чтобы CFML-код в сочетании с WML выполнялся корректно, на первой строке кода CFML-файла необходимо определять MIME-мнемокод для его содержимого следующим образом:

```
<CFCCONTENT TYPE="text/vnd.wap.wml">
```

Таким образом, для страницы устанавливается MIME-тип WML-документа — `text/vnd.wap.wml` независимо от расширения файла. В противном случае эмулятор WAP-браузера определит весь последующий код, как HTML, и при его компиляции выдаст ошибку.

Причем в ColdFusion Studio существует шаблон, позволяющий открывать новый документ с уже включенной строкой подобного характера. Выполните команду **File | New** (Файл | Создать) и в появившемся диалоговом окне **New Document** (Новый документ) во вкладке **WML** выберите шаблон **Dynamic (using CF)** (Динамический). После чего в основном окне редактирования откроется файл со следующим содержанием:

```
<CFCCONTENT TYPE="text/vnd.wap.wml">
```

```
<?xml version="1.0"?>
```

```
<!DOCTYPE wml PUBLIC "-//PHONE.COM/DTD WML 1.1//EN"
```

```
"http://www.phone.com/dtd/wml11.dtd" >
```

```
<wml>
  <card>

  </card>
</wml>
```

Данный шаблон аналогичен шаблону **Static** (Статический) с одной лишь разницей — в начале кода определен MIME-мнемокод. Что это дает? Предположим, у нас есть документ `default.cfm`, используемый сервером по умолчанию, и нам необходимо независимо от загружаемого клиента, WAP или Web, предоставлять пользователю информацию в соответствии с его устройством: телефоном или компьютером. Для решения этой задачи сформируем содержание файла `default.cfm` следующим образом:

```
<CFIF CGI.HTTP_USER_AGENT contains "wap">
  <CFCONTENT TYPE="text/vnd.wap.wml">
    <?xml version="1.0"?>
    <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
      "http://www.wapforum.org/DTD/wml_1.1.xml">
    <wml>
      <card id="start" ontimer="index.wml">
        <timer value="3"/>
        <p align="center">
        <br/><i>One moment...</i>
        </p>
      </card>
    </wml>
  <CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=index.cfm">
  </CFIF>
```

Как видно из примера, при обнаружении в переменной `CGI.HTTP_USER_AGENT` фрагмента `"wap"` выполняется код, написанный на языке WML. В противном случае загружается обычная страница `index.cfm`, подготовленная для чтения в привычном Web-браузере. На рис. 22.10 показан результат исполнения приведенного кода сразу в обоих браузерах.

Надо заметить, что файл `index.cfm` мы несколько изменили, добавив некоторое количество карт, и в качестве первой карты поместили список ссылок на существующие карты, что несколько напоминает вертикальное меню.

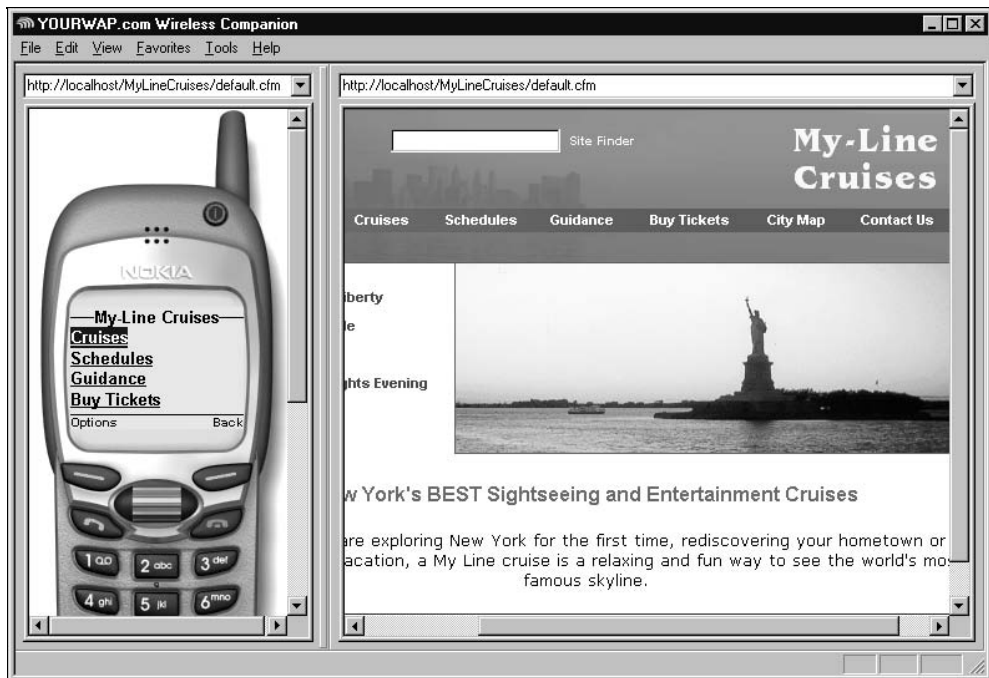


Рис. 22.10. Независимая загрузка одного и того же документа

Создание диаграммы с помощью тега `<CFGRAPH>` посредством мобильного устройства

В качестве заключительного примера представим решение следующей задачи. Предположим, что, используя мобильное устройство, нам необходимо на сервере создать CFML-файл, который бы использовался в качестве построителя диаграммы на основе таблицы `Tour` базы данных `Cruise` (зависимость цены от предлагаемых туров), с мобильного же устройства мы бы передавали заголовок и тип диаграммы, благо в ColdFusion существует тег `<CFGRAPH>`, позволяющий строить диаграммы различных типов. В частности данный тег обеспечивает построение следующих диаграмм:

- ☐ `Bar` — диаграмма в виде столбцов (гистограмма);
- ☐ `Pie` — круговая диаграмма;
- ☐ `Line` — диаграммы в виде линий (график);
- ☐ `HorizontalBar` — диаграмма в виде горизонтальных столбцов (линейчатая диаграмма).

Далее будут представлены две страницы: Graph.cfm, выступающая в качестве диалоговой формы для ввода заголовка и типа будущей диаграммы, и BuildGraph.cfm, позволяющая создать файл CostGraph.cfm с необходимым содержанием.

Листинг 22.1. Код страницы Graph.cfm

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd" >
<wml>
  <template>
    <do type="prev" name="BACK" label="Back"
      optional="false"><prev/></do>
  </template>
  <card id="Graph" title="Data input Graph">
    <onevent type="onenterforward">
      <refresh>
        <setvar name="TitleGraph" value="Cost" />
        <setvar name="TypeGraph" value="Bar" />
      </refresh>
    </onevent>
    <onevent type="onenterbackward">
      <refresh>
        <setvar name="TitleGraph" value="" />
        <setvar name="TypeGraph" value="" />
      </refresh>
    </onevent>

    <do type="Accept" label="Build">
      <go href="BuildGraph.cfm?TitleGraph=$(TitleGraph) &
        TypeGraph=$(TypeGraph) "/>
    </do>
    <p align="left" mode="nowrap">
      TitleGraph: <input type="text" name="TitleGraph"/><br/>
      TypeGraph:
      <select name="TypeGraph">
        <option value="Bar">Bar</option>
        <option value="Pie">Pie</option>
```

```

    <option value="Line">Line</option>
    <option value="HorizontalBar">HorizontalBar</option>
</select>
</p>
</card>
</wml>

```

Листинг 22.2. Код страницы BuildGraph.cfm

```

<CFCONTENT TYPE="text/vnd.wap.wml">
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd" >

<wml>
  <card id="BuildGraph" title="Build Graph">
    <CFSWITCH EXPRESSION=#TypeGraph#>
      <CFCASE VALUE="Bar">
        <CFSET Graph = '<CFGRAPH TYPE = "Bar" QUERY = "Tour"
          VALUECOLUMN = "CostAdult_tou"
          ITEMCOLUMN = "Name_tou" TITLE = "#TitleGraph#"
          SCALEFROM = "0" SCALETO = "10" DEPTH = "10"
          BARSPACING = "20" GRIDLINES = "4"
          GRAPHHEIGHT = "300" GRAPHWIDTH = "400"
          COLORLIST = "####003399,####ffcc00,####33cc00,####990033"
          ITEMLABELORIENTATION = "Horizontal"
          VALUELOCATION = "Overbar"></CFGRAPH>'>
      </CFCASE>
      <CFCASE VALUE="Pie">
        <CFSET Graph = '<CFGRAPH TYPE = "Pie" QUERY = "Tour"
          VALUECOLUMN = "CostAdult_tou"
          ITEMCOLUMN = "Name_tou" TITLE = "#TitleGraph#"
          VALUELOCATION = "Outside" BORDERWIDTH = "0"
          GRAPHHEIGHT = "450" GRAPHWIDTH = "550"
          DEPTH = "15"
          COLORLIST = "####003399,####ffcc00,####33cc00,####990033"
          SHOWLEGEND = "Right"></CFGRAPH>'>
      </CFCASE>
      <CFCASE VALUE="Line">

```

```

<CFSET Graph = '<CFGRAPH TYPE = "Line" QUERY = "Tour"
    VALUECOLUMN = "CostAdult_tou"
    ITEMCOLUMN = "Name_tou" SCALEFROM = "0"
    SCALETO = "10" TITLE = "#TitleGraph#"
    GRAPHHEIGHT = "300" GRAPHWIDTH = "400"
    DEPTH = "10" GRIDLINES = "4"></CFGRAPH>''>

</CFCASE>
<CFCASE VALUE="HorizontalBar">
    <CFSET Graph = '<CFGRAPH TYPE = "HorizontalBar" QUERY = "Tour"
        VALUECOLUMN = "CostAdult_tou"
        ITEMCOLUMN = "Name_tou" VALUELOCATION = "Overbar"
        SCALEFROM = "0" SCALETO = "10"
        TITLE = "#TitleGraph#" BARSPACING = "20"
        GRAPHHEIGHT = "300" GRAPHWIDTH = "400"
        DEPTH = "10"
        COLORLIST = "####003399,####ffcc00,####33cc00,####990033"
        GRIDLINES = "4"></CFGRAPH>''>

</CFCASE>
<CFDEFAULTCASE>
    <!-- Default Bar --->
    <CFSET Graph = '<CFGRAPH TYPE = "Bar" QUERY = "Tour"
        VALUECOLUMN = "CostAdult_tou"
        ITEMCOLUMN = "Name_tou" TITLE = "Cost"
        SCALEFROM = "0" SCALETO = "10" DEPTH = "10"
        BARSPACING = "20" GRIDLINES = "4"
        GRAPHHEIGHT = "300" GRAPHWIDTH = "400"
        COLORLIST = "####003399,####ffcc00,####33cc00,####990033"
        ITEMLABELORIENTATION = "Horizontal"
        VALUELOCATION = "Overbar"></CFGRAPH>''>

</CFDEFAULTCASE>
</CFSWITCH>

<CFFILE ACTION = "Write"
    FILE = "C:\Apache\htdocs\MyLineCruises\CostGraph.cfm"
    OUTPUT = "<CFQUERY NAME='Tour'
        DATASOURCE='#application.dsn# '>SELECT Name_tou,
        CostAdult_tou FROM Tour</CFQUERY> #Graph#">

<p>The diagram <CFOUTPUT>#TitleGraph#</CFOUTPUT>
is constructed!</p>

```

```
</card>
```

```
</wml>
```

Остается только продемонстрировать, как это работает. На рис. 22.11—22.13 показаны последовательные действия при вводе данных по формированию диаграммы.



Рис. 22.11. Загрузка диалоговой формы **Data input Graph**



Рис. 22.12. Заполнение заголовка будущей диаграммы



Рис. 22.13. Выбор типа графика

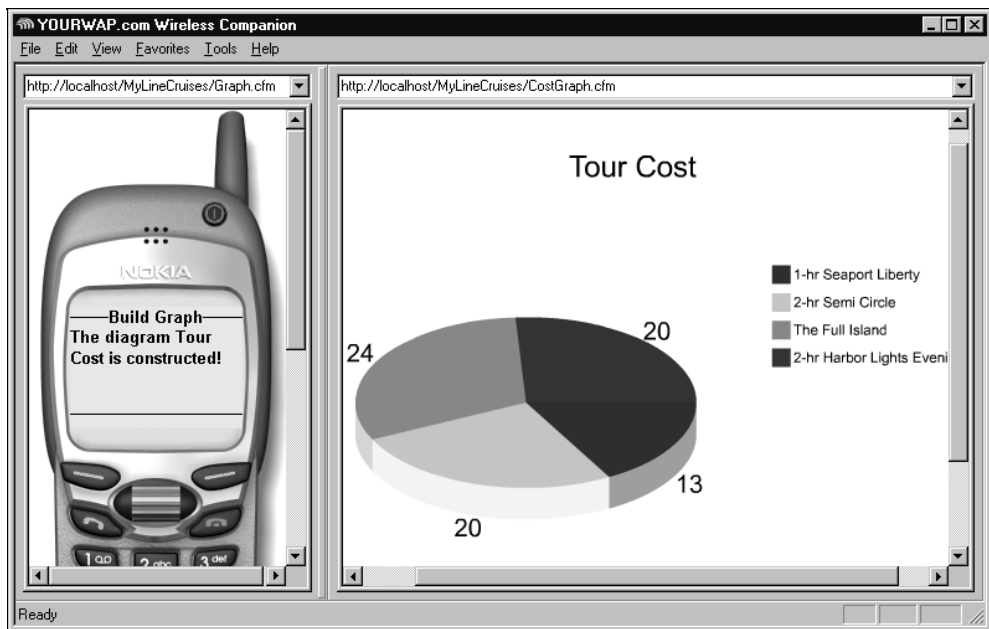


Рис. 22.14. Результат построения диаграммы

После ввода данных, как показано на предыдущих рисунках, следует выбрать пункт **Options** (Параметры) и в раскрывшемся меню выполнить ко-

манду **Build** (Построить). На рис. 22.14 продемонстрирован столь желанный результат, где страницу CostGraph.cfm необходимо просматривать с помощью стандартного Web-браузера.

Замечание

Для использования тега `<CFGRAPH>`, как мы убедились, позволяющего строить диаграммы, требуется наличие сервера Java-приложений JRun 3.0 Server, в прошлом как и ColdFusion принадлежавшего компании Allaire, а в настоящий момент являющегося программным продуктом корпорации Macromedia.

Резюме

Подведем итог. В данной главе мы использовали CFML-тег `<CFGRAPH>`.



Безопасность приложения

ColdFusion поддерживает несколько уровней безопасности.

Редакции ColdFusion Server — Professional и Enterprise — включают в себя средство Advanced Security, которое обеспечивает масштабируемую, granularную защиту для построения и разворачивания ваших приложений ColdFusion.

- ❑ Администраторы Application Development System (системы разработки приложений) могут контролировать доступ к файлам, источникам данных и управление для каждого разработчика команды. Они могут координировать использование команды на совместных серверах с гарантией защиты уязвимых данных и приложений.
- ❑ Разработчики Application Deployment ColdFusion (разработчики приложений) могут создать комплексные правила для программного контроля доступа к функциям внутри приложений. Вы можете заключить приложения в безопасных областях, таким образом гибко ограничивая доступ приложений к каталогам, компонентам, базам данных или другим ресурсам на сервере.

Безопасность Remote Development Services

Тип безопасности Remote Development Services (ColdFusion RDS security) предоставляет услуги защиты разработчикам, использующим ColdFusion Studio. RDS security лежит в основе безопасности коллективно-ориентированной среды разработки ColdFusion, в которой группам разработчиков, создающим приложения в ColdFusion Studio, требуются различные уровни доступа к файлам и ресурсам данных ColdFusion.

Когда вы работаете в ColdFusion Studio, вы обращаетесь к ресурсам дистанционно, открывая cfм-файлы, или обращаясь к источникам данных. RDS security идентифицирует вас и предоставляет вам доступ к ресурсам соответственно логике. Идентификация производится по домену Windows NT, источнику данных ODBC или протоколу LDAP (Lightweight Directory Access Protocol), определенному на странице "ColdFusion Administrator", как часть контекста защиты.

Существуют два способа применения службы RDS security:

- ❑ **Basic Security** (Основная безопасность) требует от разработчиков в ColdFusion Studio ввести пароль, после идентификации которого открывается доступ к средствам RDS security, таким как просмотр, редактирование, операции с базой данных, устранение неполадок и т. п.;
- ❑ **Advanced Security** (Повышенная безопасность) позволяет администраторам ColdFusion ограничивать или разрешать доступ к системам файлов и источникам данных, в соответствии с контекстами защиты и стратегиями, установленными в разделе **Advanced Security** (Расширенная безопасность) на странице администратора "ColdFusion Administrator".

Таким образом, будучи администратором сервера ColdFusion, вы можете задавать конфигурацию службы RDS security так, чтобы она наилучшим образом соответствовала потребностям вашей сетевой группы.

Пользовательская безопасность

Служба пользовательской защиты (User security) идентифицирует пользователей, когда они входят в приложение ColdFusion, и потом присваивает привилегии, базируясь на групповом членстве или иных определенных вами критериях. Например, допустим, что вы используете ColdFusion для построения внутренней сети вашей компании, а в последствии как ведущий узел для созданной внутренней сети. Отдел кадров поддерживает страницу в Internet, на которой все служащие могут своевременно получить информацию о компании, например о ее политике, предстоящих событиях и рабочих местах. Вы хотите, чтобы каждый мог прочесть информацию, но лишь определенные сотрудники отдела кадров могли дополнять, обновлять или удалять сведения.

Помимо этого, вы, вероятно, захотите предоставить вашим сотрудникам возможность просматривать специализированную информацию об их заработной плате, разрядах, отзывы о качестве работы. Вы, конечно, не захотите, чтобы один служащий смотрел конфиденциальную информацию о другом, но предпочтете, чтобы менеджеры могли просматривать и, возможно, обновлять сведения о своих подчиненных. Система абонентской безопасности идентифицирует и авторизует пользователей каждый раз, когда они пытаются получить доступ или работать с уязвимыми данными.

Система пользовательской защиты состоит из двух компонентов:

- ❑ контекстов защиты, заданных в разделе **Advanced Security** (Расширенная безопасность) на странице администратора "ColdFusion Administrator". Контекст защиты обеспечивает базу для идентификации и авторизации пользователей;

- программного кода, который вы формируете на страницах приложения, и который проверяет по контексту защиты, разрешен ли пользователю доступ к определенному ресурсу, и действует в зависимости от результата проверки.

Прежде чем вы сможете использовать систему пользовательской защиты в своих приложениях, вы должны убедиться, что ваш администратор ColdFusion установил повышенную безопасность на сервере и задал конфигурацию базы защиты, соответствующую вашему приложению. После того как база защиты сконфигурирована, вы можете кодировать свойства защиты в ваших приложениях ColdFusion.

Реализация расширенной безопасности на страницах приложения

Средство Advanced Security (Расширенная безопасность) облегчает разработчикам осуществление защиты приложения. После того как ваш администратор установит соответствующие контексты защиты для вашего приложения, вы сможете начать пользоваться тегами защиты ColdFusion и функциями идентификации пользователей, чтобы видеть, авторизован ли доступ в ту часть приложения, куда они пытаются войти.

Чтобы использовать теги защиты, функции идентификации и авторизации пользователей и открывать либо отказывать в доступе к ресурсам, в соответствии с правилами контекста защиты следует:

- включать тег `<CFAUTHENTICATE>` в каждое приложение, где вы хотите идентифицировать пользователей, т. е. убедиться, что они те, за кого себя выдают. Обычно тег `<CFAUTHENTICATE>` используется в конфигурационном файле `Application.cfm` вашего приложения;
- проверять идентификацию при каждом запросе, если не хотите использовать cookie. ColdFusion создает cookie `cfauth` для помещения в него информации об идентификации;
- пользоваться функцией `IsAuthenticated` для проверки идентификации текущего пользователя;
- пользоваться функцией `IsAuthorized` для проверки доступа (авторизации) пользователя к ресурсам. Эта функция позволяет разработчикам открывать или отказывать в доступе к защищенным ресурсам на основе уровня авторизации пользователя, определяемого уже установленными контекстами защиты;
- пользоваться тегом `<CFIMPERSONATE>` всякий раз, когда хотите предоставить определенному пользователю более высокий уровень доступа, чем обычно.

Кодирование страниц приложения

В качестве дополнительной меры безопасности вы можете кодировать строки в ваших приложениях при помощи функций `Encrypt` (зашифровать) и `Decrypt` (расшифровать), как это было продемонстрировано в *разд. "Шифрование пользовательских паролей" гл. 16*.

Функции идентификации и авторизации

После того как вы с помощью тега `<CFAUTHENTICATE>` проверите, определен ли данный пользователь для какого-либо контекста безопасности, вы можете использовать следующие функции защиты во всем вашем приложении в любое время, когда вам потребуется распознать или авторизовать пользователя:

- ☐ `IsAuthenticated` — проверяет, был ли текущий сеанс идентифицирован тегом `<CFAUTHENTICATE>`;
- ☐ `IsAuthorized` — проверяет, имеет ли идентифицированный пользователь доступ к указанному им ресурсу, базируясь на правилах, определенных в контексте защиты, для которого идентифицирован данный пользователь.

Функция *IsAuthenticated*

Функция `IsAuthenticated` проверяет, успешно ли выполнен тег `<CFAUTHENTICATE>` по текущему запросу. Если нет, она ищет cookie `cfauth`, чтобы определить, идентифицирован ли пользователь. Если указан атрибут `SETCOOKIE="No"` в сочетании с тегом `<CFAUTHENTICATE>`, то вы должны вызывать тег `<CFAUTHENTICATE>` при каждом запросе в приложении.

Функция `IsAuthenticated` возвращает `True`, если пользователь идентифицирован для текущего запроса, в противном случае функция возвращает `False`.

Если вызывается функция `IsAuthenticated` с произвольным параметром `security_context_name`, то возвращается значение `True`, если пользователь идентифицирован в указанном контексте защиты, в противном случае функция возвращает `False`.

Функция `IsAuthenticated` имеет следующий синтаксис:

```
IsAuthenticated([security_context_name])
```

Функция *IsAuthorized*

После идентификации пользователя вы можете вызвать функцию `IsAuthorized`, чтобы проверить, к каким ресурсам допущен данный пользо-

ватель. Вы определяете уровни авторизации, когда создаете стратегии безопасности в разделе **Advanced Security** (Расширенная безопасность) на страницы администратора "ColdFusion Administrator".

Функция `IsAuthorized` возвращает `True`, если конкретному пользователю разрешена указанная операция в заданном ресурсе ColdFusion.

Функция `IsAuthorized` включает три параметра и имеет следующий синтаксис:

```
IsAuthorized(ResourceType, ResourceName, [Action])
```

Например, чтобы проверить, разрешено ли идентифицированному пользователю производить выборки из источника данных, называемого `Cruise`, укажите:

```
IsAuthorized("Datasource", "Cruise", "Select")
```

В этом примере функция `IsAuthorized` возвращает `True`, если пользователю разрешено производить выборку из указанного источника данных, или если этот источник данных не защищен в контексте защиты.

В табл. 23.1 представлены типы ресурсов ColdFusion в соответствии с возможными действиями.

Таблица 23.1. Типы ресурсов функции `IsAuthorized`

Тип ресурса	Действия
APPLICATION	ALL USECLIENTVARIABLES
CFML	Применимые действия к тегу, указанному в качестве ResourceName
FILE	READ WRITE
DATASOURCE	ALL CONNECT SELECT INSERT UPDATE DELETE SP (от англ. <i>stored procedure</i> — хранимая процедура)
COMPONENT	Не имеет действий
COLLECTION	DELETE OPTIMIZE PURGE SEARCH UPDATE

Таблица 23.1 (окончание)

Тип ресурса	Действия
CUSTOMTAG	Не имеет действий
USEROBJECT	Действия, определяемые на странице "ColdFusion Administrator"
Function	Не имеет действий
User	Не имеет действий

Примечание

Сервер ColdFusion Server не проверяет авторизацию пользователя без специального запроса с помощью функции `IsAuthorized`. В зависимости от результатов выполнения этой функции вы сами решаете, как действовать дальше.

Тег <CFAUTHENTICATE>

Синтаксис:

```
<CFAUTHENTICATE SECURITYCONTEXT = "SecurityContext"
    USERNAME = "User_ID"
    PASSWORD = "Password"
    SETCOOKIE = "Yes | No"
    THROWONFAILURE = "Yes | No"
    AUTHTYPE = "Basic | X509">
```

В табл. 23.2 представлено описание атрибутов тега <CFAUTHENTICATE>.

Таблица 23.2. Описание атрибутов тега <CFAUTHENTICATE>

Атрибут	Описание
SECURITYCONTEXT	Контекст безопасности. Обязательный атрибут. Описывает, какой контекст безопасности нужно использовать для идентификации и авторизации. Этот контекст должен быть определен в разделе Advanced Security (Расширенная безопасность) на странице администратора "ColdFusion Administrator"
USERNAME	Имя пользователя. Обязательный атрибут для получения доступа к защищенным ресурсам
PASSWORD	Пароль пользователя. Обязательный атрибут для получения доступа к защищенным ресурсам

Таблица 23.2 (окончание)

Атрибут	Описание
SETCOOKIE	Индикатор установки cookie для помещения в нем информации об идентификации. Возможные значения: • Yes — да (по умолчанию); • No — нет. Необязательный атрибут
THROWONFAILURE	Показывает, выдавать ли ColdFusion сообщение об ошибке типа SECURITY (безопасность), если идентификация не проходит. Возможные значения: • Yes — да (по умолчанию); • No — нет. Необязательный атрибут
AUTHTYPE	Тип установления подлинности. Возможные значения: • Basic (по умолчанию); • X509. Необязательный атрибут

Например:

```
<CFAUTHENTICATE SECURITYCONTEXT = "MySecurityContext"
    USERNAME = "Anna"
    PASSWORD = "*****">
```

Если пользователь в системе еще не определен, ColdFusion выдает ошибку защиты. Вы можете или отказать в доступе к ресурсу или перенаправить пользователя на страницу регистрации. Например, вы можете вывести форму регистрации и потом, если пользователь успешно войдет в систему, вывести первоначально запрашиваемую им страницу.

Тег <CFIMPERSONATE>

Синтаксис:

```
<CFIMPERSONATE SECURITYCONTEXT = "SecurityContext"
    USERNAME = "Name"
    PASSWORD = "Password"
    TYPE = "CF | OS">
...

```

```
HTML- или CFML-код выполнения
...
</CFIMPERSONATE>
```

В табл. 23.3 представлено описание атрибутов тега <CFIMPERSONATE>.

Таблица 23.3. Описание атрибутов тега <CFIMPERSONATE>

Атрибут	Описание
SECURITYCONTEXT	Контекст безопасности. Обязательный атрибут. Описывает, какой контекст безопасности нужно использовать для идентификации и авторизации. Этот контекст может быть определен в разделе Advanced Security (Расширенная безопасность) на странице администратора "ColdFusion Administrator", если атрибут TYPE принимает значение CF, и может быть определен в домене Windows NT, соответственно, если атрибут TYPE принимает значение OS
USERNAME	Имя пользователя. Обязательный атрибут
PASSWORD	Пароль пользователя. Обязательный атрибут
TYPE	Показывает тип реализации, который нужно применить. Возможные значения: • CF — для уровня приложения; • OS — для уровня операционной системы. Реализация уровня приложения позволяет вам получить права пользователя ColdFusion по заданному контексту защиты. Реализация уровня операционной системы позволяет получить права пользователя Windows NT по заданному домену Windows NT. Реализация уровня операционной системы в настоящее время недоступно для UNIX. Обязательный атрибут

Тег <CFIMPERSONATE> предоставляет разработчикам ColdFusion способ безопасно выполнить сегмент кода. Этот тег полезен, когда вы хотите кратковременно предоставить тип доступа, в котором обычно отказываете. Допустим, вы провайдер Internet-услуг, который предоставляет услуги по разработке ColdFusion. Вы предоставляете комплект клиентских тегов, позволяющих вашим пользователям добавлять возможности, такие как счетчики обращений к странице, гостевые книги, доски объявлений в создаваемые ими приложения ColdFusion. Для предоставления таких функций вы также должны открыть доступ к некоторым ресурсам, которые предпочли бы защитить. Использование тега <CFIMPERSONATE> открывает доступ к этим функциям в безопасном режиме путем "упаковки" функций в клиентский тег.

Например, как провайдер Internet-услуг, вы конечно не захотите, чтобы ваши клиенты имели доступ к тегу <CFFILE> на ваших серверах. Однако, если вы предоставляете клиентам доску объявлений, то должны позволить им записывать собственные объявления, в системно поддерживаемые файлы, в данном случае — файл, где содержится список объявлений. Вы можете предоставить доску объявлений в пользовательском (custom) теге, в который включен тег <CFFILE>. Для гарантии, что пользовательский тег имеет доступ к тегу <CFFILE>, пользователю необходимо оказать доверительное отношение, пока выполняется тег, и вернуться назад к пользователю без оказанного доверия после того, как будет выполнен "правильный" фрагмент кода.

В следующем примере построенная конструкция с помощью тега <CFIMPERSONATE> позволяет нам в защищенный файл записать определенный текст, потому что пользователю ColdFusion по имени Anna был предоставлен доступ к файлу по контексту защиты MySecurityContext:

```
<CFIMPERSONATE SECURITYCONTEXT = "MySecurityContext"
    USERNAME = "Anna"
    PASSWORD = "****"
    TYPE = "CF">
    <CFFILE ACTION = "Write"
        FILE = "C:\Apache\htdocs\MyLineCruises\BoardAnnouncements.cfm"
        OUTPUT = "#strAnnounce#">
</CFIMPERSONATE>
```

Для выявления исключительных ситуаций безопасности вы можете использовать теги <CFTRY> и <CFCATCH>, задав тип исключительной ситуации Security тега <CFCATCH>, позволяющего выявить сбой. Для получения информации по стратегиям обработки ошибок см. главу 15.

Резюме

Подведем итог. В данной главе вы познакомились с CFML-тегами <CFAUTHENTICATE>, <CFIMPERSONATE> и функциями IsAuthenticated, IsAuthorized.



Технология Web Distributed Data Exchange

Web Distributed Data Exchange (WDDX) — это новая технология, основанная на базе расширенного языка разметки XML (Extensible Markup Language) и представленная корпорацией Allaire. Применение модели WDDX облегчает компаниям обмен структурированными данными между Web-приложениями. WDDX дает компаниям возможность создать "Web Syndicate Networks" (Сетевые Web-синдикаты), где содержимое и коммерческие активы любого Web-сайта могут быть представлены как услуги другим Web-сайтам. Эта технология делает возможными появление и развитие новых видов бизнеса в Web и увеличивает способность к взаимодействию между различными прикладными платформами.

В простейшей своей форме WDDX может использоваться для обмена данными без необходимости непосредственного применения XML. Это позволяет разработчикам легко создавать приложения типа "браузер-сервер" для связки данных, подобные традиционным системам клиент-сервер, автономным Web-приложениям и распределенным приложениям типа "сервер-сервер". Наибольшие возможности открываются при использовании технологии для объединения приложений в так называемые "синдикаты". С помощью WDDX Web-сайт, построенный на основе Perl, может легко обмениваться содержимым баз данных и представить все операции и процедуры на Web-сайте, созданном с помощью платформы типа ColdFusion, ASP или других серверов Web-приложений.

"WDDX обеспечивает богатую основу для объединения в "синдикаты", — сказал Джереми Эллэйр (Jeremy Allaire), вице-президент по стратегии технологии, в настоящий момент занимающий пост технического директора корпорации Macromedia. — Поскольку тысячи организаций используют Web, и новые компании появляются вокруг Web бизнес-модели, жизненно важно, чтобы приложения, разработанные для поддержки on-line бизнеса, могли бы легко управлять активами приложения других компаний, независимо от платформы, на которой они были созданы. Через такую способность к взаимодействию Web станет основой для новых бизнес-моделей, основанных на сетевых "синдикатах".

"WDDX решает самые серьезные проблемы, связанные с обменом данными между различными средами Web-программирования, — сказал Г. Хуссейн Чиной (G. Hussain Chinoy), CEO и главный информационный менеджер компании Granularity Information Architecture, Inc. — С помощью WDDX мы легко можем создавать приложения для электронной коммерции и EDI-систем, использующих любую платформу сервера приложения или инструмент разработки, будь то ColdFusion, ASP, Perl или Java, и зная, что наш вклад может управляться любым другим разработчиком на любой другой платформе. WDDX открывает дверь прямо в мир языков программирования."

Распределение и совместимость платформ

WDDX — это бесплатная, открытая технология, доступная в виде комплекта для разработки обеспечения (SDK) на спонсируемом корпорацией Allaire Web-сайте WDDX.org (OpenWDDX.org), созданном, чтобы облегчить постоянное развитие и распространение WDDX. WDDX SDK доступен для любой платформы, которая поддерживает XML и популярные языки Web-программирования. SDK включает в себя библиотеки, образец кода, справочные материалы и документацию. И Web-сайт, и SDK были созданы, чтобы сделать возможным взаимодействие между Web-приложениями, и способствовать появлению новых бизнес-моделей на основе объединенных Web-сетей.

"WDDX решает целый ряд проблем, знакомых опытным разработчикам Web-приложений, таких как: распределение приложений, хранение комплексных данных и обмен данными при использовании различных языков программирования, — сказал Бен Арчибалд (Ben Archibald), основатель компании ChangeMedia. — В ChangeMedia мы поняли захватывающую эффективность разработки путем хранения комплексных объектов данных в родственных таблицах или простых файлах. Используя WDDX, мы можем избежать традиционного анализа при повышении мобильности информации между SQL-платформами."

Тег <CFWDDX>

Синтаксис:

```
<CFWDDX ACTION = "CFML2WDDX | WDDX2CFML | CFML2JS | WDDX2JS"  
  INPUT = "Inputdata"  
  OUTPUT = "ResultVariableName"  
  TOPLEVELVARIABLE = "TopLevelVariableNameforJavaScript"  
  USETIMEZONEINFO = "Yes | No"  
  VALIDATE = "Yes | No">
```

В табл. 24.1 представлено описание атрибутов тега <CFWDDX>.

Таблица 24.1. Описание атрибутов тега <CFWDDX>

Атрибут	Описание
ACTION	Действие. Возможные значения: - CFML2WDDX — сериализует из CFML- в WDDX-формат; - WDDX2CFML — десериализует из WDDX- в CFML-формат; - CFML2JS — сериализует из CFML- в JavaScript-формат; - WDDX2JS — десериализует из WDDX в JavaScript. Обязательный атрибут
INPUT	Значение, представленное для обработки. Обязательный атрибут
OUTPUT	Наименование переменной, которой присваивается результат выполнения действия. Обязательный атрибут для действия WDDX2CFML, во всех остальных действиях при отсутствии определения данного атрибута результат выполнения тега будет выведен в поток HTML
TOplevelVariable	Наименование объекта JavaScript, создаваемого определяемым процессом. Обязательный атрибут только для действий CFML2JS и WDDX2JS
USETimezoneInfo	Индикатор принятия во внимание информации часового пояса. Возможные значения: - Yes — да (по умолчанию); - No — нет. Необязательный атрибут
VALIDATE	Индикатор выполнения проверки ошибок. Возможные значения: - Yes — да; - No — нет (по умолчанию). Необязательный атрибут

Пример использования WDDX

Идея WDDX состоит в том, чтобы преобразовать (сериализовать) любую стандартную структуру данных: целое число, строку, массив, запрос и т. д. — в представление в формате XML, соответствующее тому или иному определению типов документов. Затем эти данные можно переслать по любому протоколу связи, способному передавать документы XML: FTP, HTTP,

SMTP и т. п. В свою очередь, данные могут быть обратно преобразованы (десериализованы) из формата WDDX в исходную структуру. При этом типы преобразованных данных остаются неизменными, если вы сериализуете запрос или массив в пакет WDDX, то при обратном преобразовании полученные данные примут тип запроса или массива соответственно. Например:

```
<HTML>
<HEAD>
    <TITLE>WDDX Query</TITLE>
</HEAD>
<BODY>
    <!-- Создаем запрос -->
    <CFQUERY NAME="QueryTour" DATASOURCE="#application.dsn#">
        SELECT Name_tou, NumLandingPlaces_tou,
               Time_tou, CostAdult_tou
        FROM Tour
    </CFQUERY>

    <!-- Сериализация данных из CFML- в WDDX-формат -->
    <CFWDDX ACTION = "CFML2WDDX"
        INPUT = #QueryTour#
        OUTPUT = "varWDDX">

    <i>WDDX XML packet:</i><br>
    <table width="100%">
        <tr><td><CFOUTPUT>#varWDDX#</CFOUTPUT></td></tr>
    </table>

    <!-- Десериализация из WDDX в CFML -->
    <CFWDDX ACTION = "WDDX2CFML"
        INPUT = #varWDDX#
        OUTPUT = "DeQueryTour">

    <br><i>Deserializing WDDX packet:</i>
    <CFTABLE QUERY="DeQueryTour" COLHEADERS="Yes"
        COLSPACING="0" HEADERLINES="1">
        <CFCOL HEADER = "<b>Name tour</b>" WIDTH = "27" ALIGN = "Left"
            TEXT = "#Name_tou#">
        <CFCOL HEADER = "<b>Places</b>" WIDTH = "5" ALIGN = "Right"
```

```

TEXT = "#NumLandingPlaces_tou#">
<CFCOL HEADER = "<b>Time</b>" WIDTH = "9" ALIGN = "Right"
TEXT = "#Time_tou#">
<CFCOL HEADER = "<b>Adults</b>" WIDTH = "9" ALIGN = "Right"
TEXT = "#CostAdult_tou#">
</CFTABLE>
</BODY>
</HTML>

```

В приведенном примере создается запрос `QueryTour`, впоследствии сериализованный в пакет WDDX, и затем уже десериализованный в объект запроса `DeQueryTour`. На рис. 24.1 изображен результат выполнения приведенного выше примера.

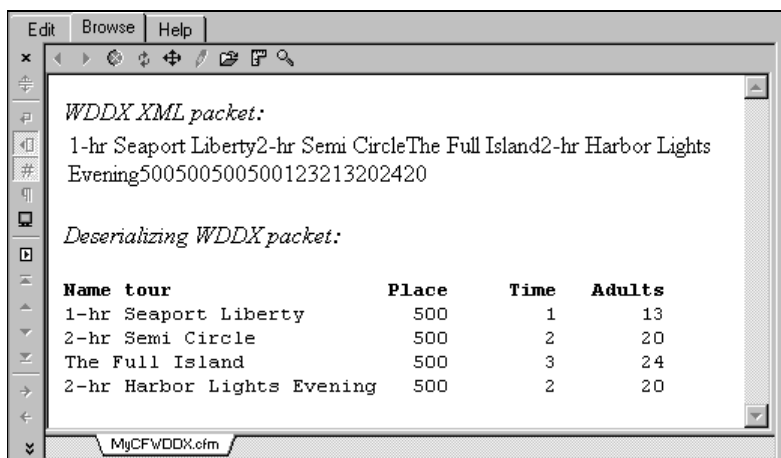


Рис. 24.1. Результат сериализации из CFML в WDDX и обратной десериализации в CFML

Таким же образом вы можете преобразовывать данные в JavaScript как из CFML, так и из WDDX, используя при этом с тегом `<CFWDDX>` такие действия, как `CFML2JS` и `WDDX2JS`.

Языковая поддержка WDDX

WDDX SDK обеспечивает модули для JavaScript 1.x, ColdFusion, COM, Perl и Java. С модулем COM, WDDX может использоваться внутри Active Server Pages, Visual Basic, Delphi, Java, PowerBuilder, C++ и Microsoft Office.



Настройка ColdFusion-расширений для Macromedia Dreamweaver

В программном продукте Dreamweaver от корпорации Macromedia, с которой уже "побратался" Allaire — фирма — разработчик ColdFusion, существует возможность подключать различные расширения от сторонних разработчиков, что позволяет в свою очередь использовать дополнительные инструменты для создания Web-сайтов. Так вот и для ColdFusion существует подобное расширение. На текущий момент времени это расширение представляет собой файл MX15666_AllaireColdFusion.mxp.

ColdFusion-расширение для Macromedia Dreamweaver может быть найдено в Internet по адресу <http://dynamic.macromedia.com/bin/MM/exchange/dreamweaver/main.jsp>.

Вам также будет необходимо установить программный продукт Macromedia Extension Manager, позволяющий подключать расширения для Dreamweaver.

Для подключения расширения ColdFusion необходимо выполнить следующие действия:

1. Запустить программу Extension Manager.
2. Из раскрывающегося списка в панели инструментов выбрать версию Dreamweaver, по отношению к которой будут применены последующие действия.

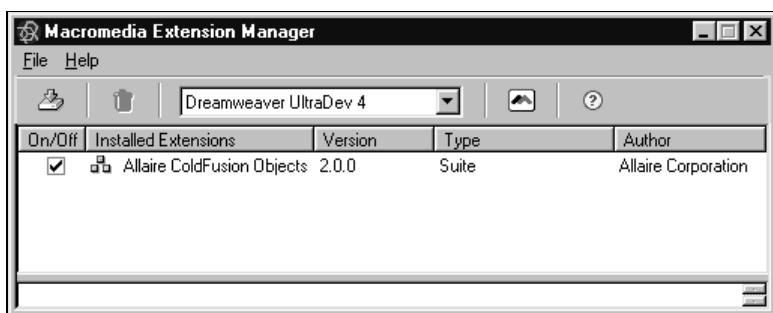


Рис. 25.1. Macromedia Extension Manager

3. Выполнить команду **File | Install Extension** (Файл | Установить расширение) или нажать комбинацию клавиш <Ctrl>+<I>.
4. Выбрать предварительно скопированный файл-расширение ColdFusion (MX15666_AllaireColdFusion.mxp).
5. Нажать на кнопку **Install** (Установить).

На рис. 25.1 показан результат установки расширения ColdFusion для Dreamweaver 4.

Использование расширения ColdFusion

ColdFusion-расширения позволяют включать функции CFML непосредственно в редакторе Dreamweaver с помощью главного меню **Insert | Allaire** (Вставка | Allaire) (рис. 25.2) или панели **Objects** (Объекты) (рис. 25.3).

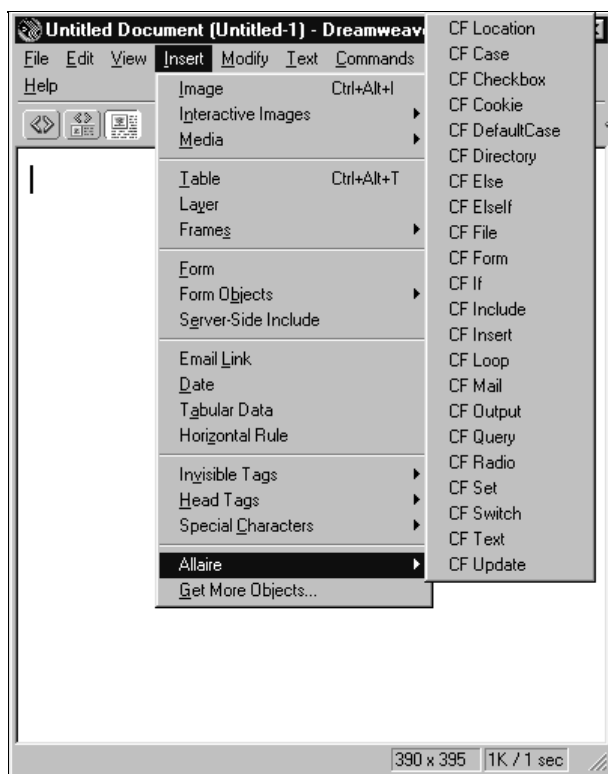


Рис. 25.2. Вставка CFML-тегов с помощью главного меню

На рис. 25.2 приведен перечень тегов ColdFusion, о которых было достаточно рассказано в предыдущих главах. После установки расширения ColdFusion в списке панели **Objects** (Объекты) появится новый набор инструментов под названием **Cold Fusion**, выбрав который, можно с легкостью вставлять CFML-теги, редактируя очередную Web-страницу.

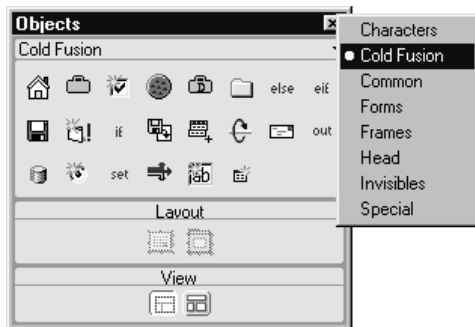


Рис. 25.3. Вставка CFML-тегов с помощью панели **Objects**

При выборе одного из этих инструментов предлагается настроить параметры соответствующего тега в режиме диалога. В качестве примера давайте разберем инструмент **CFLocation** (соответствующий тег вставляется на страницу командой **Insert | Allaire | CF Location**), расположенный на панели в виде "домика"  и предназначенный для вставки тега `<CFLLOCATION>`, способствующего открытию указанного ресурса. На рис. 25.4 изображено диалоговое окно редактирования параметров тега `<CFLLOCATION>`.

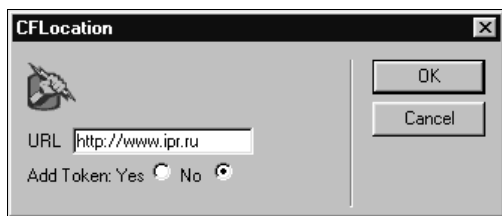


Рис. 25.4. Диалоговое окно редактирования параметров тега <CFLOCATION>

После того как тег создан, вы можете редактировать его параметры в инспекторе **Properties** (Свойства) Dreamweaver (рис. 25.5).

Также, щелкая на значке **Quick Tag Editor**  в окне свойств, вы можете редактировать код CFML непосредственно в дополнительном окне (рис. 25.6).

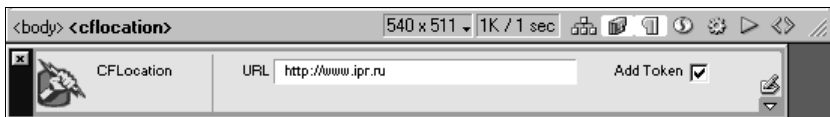


Рис. 25.5. Инспектор **Properties** Dreamweaver

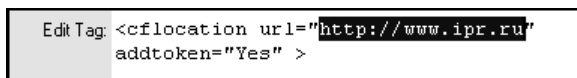


Рис. 25.6. Редактирование CFML-кода в дополнительном окне

У Macromedia существует еще один продукт — Dreamweaver UltraDev, позволяющий таким же образом подключать расширения для ColdFusion. При этом в Dreamweaver UltraDev представлено средство **Data Bindings** (рис. 25.7), позволяющее разрабатывать приложения, основанные на ColdFusion.

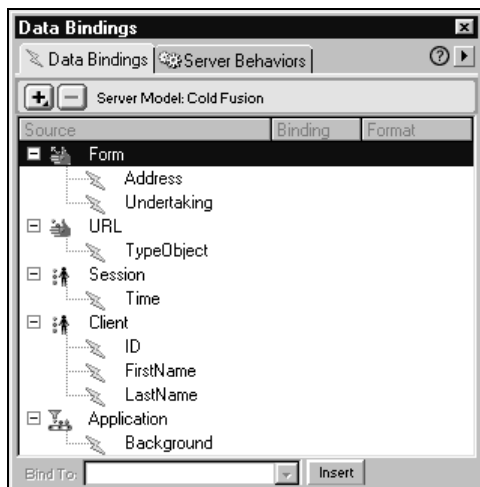


Рис. 25.7. Окно инструмента Data Bindings в Dreamweaver UltraDev

Здесь вы можете устанавливать доступ к базам данных для дальнейшей работы с колонками таблиц. Также вы можете определять переменные форм, URL, сессий, клиентские переменные и переменные приложений. Что позволяет в полной мере облегчить работу по созданию Web-страниц с использованием языка ColdFusion.

Приложение 1



Коды основных файлов проекта "My-Line Cruises"

Данное приложение является дополнением к материалу глав, представляющих описание проекта "My-Line Cruises", и предназначено для демонстрации кода основных файлов данного проекта, в частности: main.css, main_ie.css, main.js, index.cfm, MyHEAD.cfm, BasicBlock.cfm.

Листинг П1.1. Код файла main.css

```
.dialog {
    cursor : hand;
    text-decoration : none;
    padding : 3px;
    font-family : Arial;
    font-size : 11px;
}

.dialog_ {
    text-decoration : none;
    padding : 3px;
    font-family : Arial;
    font-size : 11px;
}

.title {
    color : #666699;
    font-family : 'Arial';
    font-size : 16px;
    padding : 0px;
    text-decoration : none;
    font-weight : bold;
}

.txt {
    color : #000000;
```

```
text-decoration : none;
padding : 0px;
font-weight : normal;
font-family : 'Verdana, Geneva, Arial, Helvetica, sans-serif';
font-size : 15px;
}

.txtb {
color : #000000;
text-decoration : none;
padding : 0px;
font-weight : bold;
font-family : 'Verdana, Geneva, Arial, Helvetica, sans-serif';
font-size : 15px;
}

A {color:#122C6D;}
A:hover {color: #3366CC;}
#imgt { position:absolute; left:0; top:0; width:650; height:99;
margin:0;}
#tabml { position:absolute; left:100; top:79; width:500; height:16;
margin:0;z-index:2;}

.menact {
color : #FFFFFF;
font-family : 'Arial';
font-size : 11px;
padding : 3px;
text-decoration : underline;
font-weight : bold;
}

.men {
color : #FFFFFF;
font-family : 'Arial';
font-size : 11px;
padding : 3px;
text-decoration : none;
font-weight : bold;
}

.menleft {
color : #4B4558;
font-family : 'Arial';
```

```
font-size : 11px;
padding : 3px;
text-decoration : none;
font-weight : bold;
}

.menleft_u {
color : #4B4558;
font-family : 'Arial';
font-size : 11px;
padding : 3px;
font-weight : bold;
}

.titletable {
color : #FFFFFF;
text-decoration : none;
padding : 0px;
font-weight : normal;
font-family : 'Arial, Helvetica, sans-serif';
font-size : 12px;
}

.itemtable {
color : #000000;
text-decoration : none;
padding : 0px;
font-weight : normal;
font-family : 'Arial, Helvetica, sans-serif';
font-size : 13px;
}

.submen1 {
color : #FFFFFF;
font-family:'Arial';
font-weight:bold;
font-size:10px;
padding:3px;
text-decoration:none;
height:19;
width: 130;
z-index : 3;
}
```

```
.submen2 {
    color : #FFFFFF;
    font-family:'Arial';
    font-weight:bold;
    font-size:10px;
    padding:3px;
    text-decoration:none;
    height:19;
    width: 80;
    z-index : 3;
}

.FindGo {
    color : #FFFFFF;
    font-family:'Arial';
    font-weight:bold;
    font-size:10px;
    padding:3px;
    text-decoration:none;
    height:19;
    width: 80;
    z-index : 3;
}

#mm0 { position:absolute; left:-300; top:90; width:6; z-index:3;}
#tablemm0 { background:#605870;}
#mm1 { position:absolute; left:-300; top:90; width:132; z-index:3;}
#tablemm1 { background:#605870;}
#mm1_1 { position:absolute; left:1; top:1;background:#605870;}
#mm1_2 { position:absolute; left:1; top:20;background:#605870;}
#mm1_3 { position:absolute; left:1; top:39;background:#605870;}
#mm2 { position:absolute; left:-300; top:100; width:100; z-index:3;}
#mm3 { position:absolute; left:-300; top:100; width:100; z-index:3;}
#mm4 { position:absolute; left:-300; top:100; width:100; z-index:3;}
#mm5 { position:absolute; left:-300; top:100; width:100; z-index:3;}
#mm6 { position:absolute; left:-300; top:100; width:70; z-index:3;}
#tablemm6 { background:#605870;}
#mm6_1 { position:absolute; left:1; top:1;background:#605870;}
#mm6_2 { position:absolute; left:1; top:20;background:#605870;}
#mm6_3 { position:absolute; left:1; top:39;background:#605870;}
```

```
.smalltxt {
    color : Black;
    text-decoration : none;
    padding : 3px;
    font-weight : normal;
    font-family : "Arial";
    font-size : 10px;
}

BODY {
    text-decoration : none;
}

.txtForm1 {
    padding : 0px;
    font-weight : normal;
    font-family : Verdana, Geneva, Arial;
    font-size : 11px;
    color : Black;
}

.txtForm2 {
    padding : 0px;
    font-weight : normal;
    font-family : Verdana, Geneva, Arial;
    font-size : 11px;
    color : White;
}

.txtTab {
    padding : 0px;
    font-weight : normal;
    font-family : Verdana, Geneva, Arial;
    font-size : 10px;
    color : Black;
}

.txtTitleTab {
    padding : 0px;
    font-weight : bold;
    font-family : Verdana, Geneva, Arial;
    font-size : 11px;
    color : Black;
}
```

```
.txtError {
    padding : 0px;
    font-weight : normal;
    font-family : Verdana, Geneva, Arial;
    font-size : 11px;
    color : Maroon;
}

INPUT {
    padding : 0px;
    font-weight : normal;
    font-family : Verdana, Geneva, Arial;
    font-size : 11px;
}

TEXTAREA {
    padding : 0px;
    font-weight : normal;
    font-family : Verdana, Geneva, Arial;
    font-size : 11px;
}

#hom { position:absolute; left:638; top:81; width:16; height:15;
        margin:0;z-index:2;}

#txt1 { position:absolute; left:50; top:130; width:659; height:10;
        margin:0;z-index:2;}
```

Листинг П1.2. Код файла main_ie.css

```
.title {
    color : #666699;
    font-family : 'Arial';
    font-size : 16px;
    padding : 0px;
    text-decoration : none;
    font-weight : bold;
}

.txt {
    color : #000000;
    text-decoration : none;
    padding : 0px;
    font-weight : normal;
```



```
font-family : 'Verdana, Geneva, Arial, Helvetica, sans-serif';
font-size : 13px;
}

.txtb {
color : #000000;
text-decoration : none;
padding : 0px;
font-weight : bold;
font-family : 'Verdana, Geneva, Arial, Helvetica, sans-serif';
font-size : 13px;
}

.titletable {
color : #FFFFFF;
text-decoration : none;
padding : 0px;
font-weight : normal;
font-family : 'Arial, Helvetica, sans-serif';
font-size : 11px;
}

.itemtable {
color : #000000;
text-decoration : none;
padding : 0px;
font-weight : normal;
font-family : 'Arial, Helvetica, sans-serif';
font-size : 11px;
}
```

Листинг П1.3. Код файла main.js

```
var ns = (document.layers)? true:false
var ie = (document.all)? true:false
var IDTimer_cir = null;
version = {num:navigator.appVersion};

colorschem =
{
gm_txt_act:'#CBCCBD',
gm_txt_deact:'#FFFFFF',
```

```

    qm_bg_act:'#4B4558',
    sm_txt_act:'#CBCCBD',
    sm_txt_deact:'#FFFFFF',
    sm_bg_act:'#4B4558',
    menu_bg_deact: '#605870'
};

var IDTimer_m = null;
var mm0 = '';
var mm1 = '';
var mm2 = '';
var mm3 = '';
var mm4 = '';
var mm5 = '';
var mm6 = '';
var activnum = 0;

function init(loadParam){
    if (ns) {document.layers['tabm1'].top = 76;}
    if (loadParam > -1) {
        if(ns){
            for (var i=0; i<7; i++) {
                linecommand_ = "mm"+i+" = document.layers['mm'+i+'"]";
                eval(linecommand_);
                linecommand_topmm = "mm"+i+".top = 100";
                eval(linecommand_topmm);
            }
        } else {
            for (var i=0; i<7; i++) {
                linecommand_ = "mm"+i+" = document.all['mm'+i+'"].style";
                eval(linecommand_);
                ie5 = (version.num.indexOf('MSIE 5')>0)
                if (ie5) {
                    linecommand_topmm = "mm"+i+".top = 99";
                    eval(linecommand_topmm);
                }
            }
        }
    }
}

```

```
function mouse_over(id, subnum, position){
    if (ie){
        if (!activnum==0) {
            linecommand1 =
"document.all['m1_'+activnum+''].style.color=colorschem.gm_txt_deact";
            linecommand2 =
"document.all['m1_'+activnum+''].style.background=document.all['tdtabm1']
                .style.background";

            eval(linecommand1);
            eval(linecommand2);
        }
        document.all[id].style.color=colorschem.gm_txt_act;
        document.all[id].style.background=colorschem.qm_bg_act;
    }

    if (!activnum==0) {
        linecommand = "mm"+activnum+".left = -300";
        eval(linecommand);
    }

    activnum = subnum;
    if (!position==0) {
        linecommand = "mm"+activnum+".left = "+position;
        eval(linecommand);
    }
}

function mouse_out(subnum){
    linecommand = "IDTimer_m =
        window.setTimeout('tool_goOut_m(\"+subnum+\"')', 60)";
    eval(linecommand);
}

function MenuOver_menu(name, id){
    if(ie){
        document.all[name].style.color=colorschem.sm_txt_act;
        document.all[id].style.background=colorschem.sm_bg_act;
    }
    clearTimeout(IDTimer_m);
}
```

```
function MenuOut_menu(name, id, subnum){
    if(ie){
        document.all[name].style.color=colorschem.sm_txt_deact;
        document.all[id].style.background=colorschem.menu_bg_deact;
    }
    linecommand = "IDTimer_m =
        window.setTimeout('tool_goOut_m(\"+subnum+)\", 600)";
    eval(linecommand);
}

function tool_goOut_m(subnum) {
    linecommand = "mm"+subnum+".left = -300";
    eval(linecommand);
    if(ie){
        linecommand1 =
"document.all['m1_"+subnum+"'].style.color=colorschem.gm_txt_deact";
        linecommand2 =
"document.all['m1_"+subnum+"'].style.background=document.all['tdtabm1']
        .style.background";
        eval(linecommand1);
        eval(linecommand2);
    }
}

function m1_click(lnk) {
    window.location.href = lnk;
}

function OpenWind(file, namewind, t, l, w, h){
    linecommand = "Variable=window.open('"+file+"', '"+namewind+"',
        ' top="+t+", left="+l+", width="+w+", height="+h+"')";
    eval(linecommand);
}

function mouse_over_bull(name, parent){
    if (ns) document[parent].document.images[name].src="bullet2.gif";
    else if (ie) document.images[name].src="bullet2.gif";
}
```

```

function mouse_out_bull(name, parent){
    if (ns) document[parent].document.images[name].src="bullet.gif";
    else if (ie) document.images[name].src="bullet.gif";
}

function del(url) {
    if(ie){
        _del = self.showModalDialog ("DialogQuestion.cfm?text_=You really
        want to delete the current record?", null,
        "status:no;center:yes;help:no;minimize:no;maximize:no;border:thin;
        statusbar:no;dialogWidth:305px;dialogHeight:110px");
        if(_del==true) {
            location.href=url;
        }
    } else {
        location.href=url;
    }
}

```

Листинг П1.4. Код страницы index.cfm

```

<HTML>
<HEAD>
    <CFINCLUDE TEMPLATE = "MyHEAD.cfm">
    <link rel="STYLESHEET" type="text/css" href="main.css">
    <script type="text/javascript" language="JavaScript"
        src="main.js"></script>
    <style type="text/css">
        <!--
        #fotoNY { position:absolute; left:219; top:124; width:431;
            height:153; margin:0;z-index:0;}
        #textleft { position:absolute; left:20; top:140; width:200;
            height:100; margin:0;z-index:0;}
        #textcentr { position:absolute; left:20; top:300; width:630;
            height:100; margin:0;}
        -->
    </style>
    <script language="JavaScript">
        if (ie) {document.write('<link rel="stylesheet"
            type="text/css" href="main_ie.css" title="master">');}
    }

```

```

</script>
</HEAD>
<BODY onload="init(1)"
    bgcolor="#CFOUTPUT>#application.BGColorPage#</CFOUTPUT>"
    text="Black" alink="#107098" link="#0C1C47" vlink="#003366">
<CFINCLUDE TEMPLATE = "BasicBlock.cfm">
<DIV id="fotoNY">
    <IMG src="fotoNY.gif" width="431" height="153" alt="" border="0">
</DIV>
<DIV id="textleft">
<TABLE id="tableft" border="0" width="200" height="100"
    cellspacing="0" cellpadding="0">
    <TR><TD><a href="SightseeingCruises.cfm#1"
        onMouseOver=" mouse_over_bull('bull1', 'textleft');
        window.status='1-hr Seaport Liberty'; return true"
        onMouseOut="mouse_out_bull('bull1', 'textleft');
        window.status=''; return true" class="menleft">
        <IMG name="bull1" src="bullet.gif" width="20" height="20"
            border="0"></a></TD>
    <TD><a href="SightseeingCruises.cfm#1"
        onMouseOver=" mouse_over_bull('bull1', 'textleft');"
        onMouseOut=" mouse_out_bull('bull1', 'textleft');"
        class="menleft">1-hr Seaport Liberty</a></TD></TR>
    <TR><TD><a href="SightseeingCruises.cfm#2"
        onMouseOver=" mouse_over_bull('bull2', 'textleft');
        window.status='2-hr Semi Circle'; return true"
        onMouseOut="mouse_out_bull('bull2', 'textleft');
        window.status=''; return true" class="menleft">
        <IMG name="bull2" src="bullet.gif" width="20" height="20"
            border="0"></a></TD>
    <TD><a href="SightseeingCruises.cfm#2"
        onMouseOver=" mouse_over_bull('bull2', 'textleft');"
        onMouseOut=" mouse_out_bull('bull2', 'textleft');"
        class="menleft">2-hr Semi Circle</a></TD></TR>
    <TR><TD><a href="SightseeingCruises.cfm#3"
        onMouseOver=" mouse_over_bull('bull3', 'textleft');
        window.status='3-hr Full Island'; return true"
        onMouseOut="mouse_out_bull('bull3', 'textleft');
        window.status=''; return true" class="menleft">

```

```

        <IMG name="bull13" src="bullet.gif" width="20" height="20"
            border="0"></a></TD>
    <TD><a href="SightseeingCruises.cfm#3"
        onMouseOver=" mouse_over_bull('bull13', 'textleft');"
        onMouseOut=" mouse_out_bull('bull13', 'textleft');"
        class="menleft">3-hr Full Island</a></TD></TR>
<TR><TD><a href="HarborLightsEvening.cfm"
    onMouseOver=" mouse_over_bull('bull14', 'textleft');"
    window.status='2-hr Harbor Lights Evening'; return true"
    onMouseOut="mouse_out_bull('bull14', 'textleft');"
    window.status=''; return true" class="menleft">
    <IMG name="bull14" src="bullet.gif" width="20" height="20"
        border="0"></a></TD>
    <TD><a href="HarborLightsEvening.cfm"
        onMouseOver=" mouse_over_bull('bull14', 'textleft');"
        onMouseOut=" mouse_out_bull('bull14', 'textleft');"
        class="menleft">2-hr Harbor Lights Evening</a></TD></TR>
<TR><TD><a href="SpecialEvents.cfm"
    onMouseOver=" mouse_over_bull('bull15', 'textleft');"
    window.status='Special Events'; return true"
    onMouseOut="mouse_out_bull('bull15', 'textleft');"
    window.status=''; return true" class="menleft">
    <IMG name="bull15" src="bullet.gif" width="20" height="20"
        border="0"></a></TD>
    <TD><a href="SpecialEvents.cfm"
        onMouseOver=" mouse_over_bull('bull15', 'textleft');"
        onMouseOut=" mouse_out_bull('bull15', 'textleft');"
        class="menleft">Special Events</a></TD></TR>
</TABLE>
</DIV>
<DIV id="textcentr">
<TABLE border="0" width="610" cellspacing="0" cellpadding="2">
    <TR>
        <TD class="title" colspan="2" align="center">
            New York's BEST Sightseeing and Entertainment Cruises<br><br>
        </TD>
    </TR>
    <TR>
        <TD class="txt" colspan="2" align="center">

```

Whether you are exploring New York for the first time, rediscovering your hometown or seeking a mini-vacation, a My Line cruise is a relaxing and fun way to see the world's most famous skyline.

```
</TD>
```

```
</TR>
```

```
</TABLE>
```

```
<br>
```

```
<TABLE border="0" width="610" cellspacing="0" cellpadding="0">
```

```
<TR><TD colspan="2" bgcolor="#122c6d" height=1></TD></TR>
```

```
</TABLE>
```

```
<TABLE border="0" width="610" cellspacing="0" cellpadding="2">
```

```
<TR>
```

```
<TD colspan="2" class="smalltxt" align="center">
```

```
<br>There the data of Circle-Line Cruises company are used  
(usage is authorized&nbsp;ns;
```

```
<a href="mailto:Ruben@ipr.ru">Ruben Akhayan</a><br>
```

```
</TD>
```

```
</TR>
```

```
</TABLE>
```

```
</DIV>
```

```
</BODY>
```

```
</HTML>
```

Листинг П1.5. Код страницы MyHEAD.cfm

```
<meta name="GENERATOR" content="ColdFusion Studio">
<meta name="Author" content="Ruben Akhayan">
<meta http-equiv="Description" content="My-Line Cruises">
<meta http-equiv="Keywords" content="Cruises">
<title>My-Line Cruises</title>
```

Листинг П1.6. Код страницы BasicBlock.cfm

```
<style type="text/css">
<!--
#FindBlock {position:absolute; left:165; top:12; width:300;
height:20; margin:0;z-index:2;}
-->
```



```
</style>
<script language="JavaScript">
    function SiteFind() {
        if(!ie) var textF = document.FindForm.TextFind.value;
        else    var textF =
            document.layers['FindBlock'].document.FindForm.TextFind.value;
        if (textF=="") {
            alert('No text Find');
        } else {
            eval("self.location.replace('Find.cfm?criterion="+textF+"')");
        }
    }
}
</script>

<DIV id="imgt">
    <p><IMG name="imgt_" src="title.gif" border="0"></p>
</DIV>

<DIV id="FindBlock">
    <FORM name="FindForm" onSubmit="SiteFind(); return false">
        <TABLE border="0"><TR>
            <TD><input type="text" name="TextFind"
                value="<CFIF isDefined('URL.Criterion')>
                <CFOUTPUT>#URL.Criterion#</CFOUTPUT></CFIF>" size="20">
            </TD>
            <TD><a HREF="javascript:SiteFind()"
                onMouseOver="window.status='Site Finder'; return true"
                onMouseOut="window.status=''; return true"
                class="FindGo">Site Finder</a></TD>
        </TR></TABLE>
    </FORM>
</DIV>

<DIV id="mm0">
    <TABLE id="tablemm0" border="0" width="6" height="0"
        cellspacing="0" cellpadding="0">
        <TR><TD></TD></TR>
    </TABLE>
</DIV>
```

```

<DIV id="mm1">
  <TABLE id="tablemm1" border="0" width="132" height="60"
    cellspacing="0" cellpadding="0">
    <TR><TD id="mm1_1"><a href="SightseeingCruises.cfm" name="mm1_1"
      onMouseOver="MenuOver_menu('mm1_1','mm1_1');
        window.status='Sightseeing Cruises';
        return true"
      onMouseOut="MenuOut_menu('mm1_1','mm1_1', 1);
        window.status='';return true"
      class="submen1">&nbsp;Sightseeing Cruises</a>
    </TD></TR>
    <TR><TD id="mm1_2"><a href="HarborLightsEvening.cfm" name="mm1_2"
      onMouseOver="MenuOver_menu('mm1_2', 'mm1_2');
        window.status='Harbor Lights Evening';
        return true"
      onMouseOut="MenuOut_menu('mm1_2', 'mm1_2', 1);
        window.status=''; return true"
      class="submen1">&nbsp;Harbor Lights Evening</a>
    </TD></TR>
    <TR><TD id="mm1_3"><a href="SpecialEvents.cfm" name="mm1_3"
      onMouseOver="MenuOver_menu('mm1_3', 'mm1_3');
        window.status='Special Events'; return true"
      onMouseOut="MenuOut_menu('mm1_3', 'mm1_3', 1);
        window.status=''; return true"
      class="submen1">&nbsp;Special Events</a>
    </TD></TR>
  </TABLE>
</DIV>
<DIV id="mm2">
  <TABLE id="tablemm2" border="0" width="100" height="0"
    cellspacing="0" cellpadding="0">
    <TR><TD></TD></TR>
  </TABLE>
</DIV>
<DIV id="mm3">
  <TABLE id="tablemm3" border="0" width="100" height="0"
    cellspacing="0" cellpadding="0">
    <TR><TD></TD></TR>
  </TABLE>

```

```
</DIV>
<DIV id="mm4">
  <TABLE id="tablemm4" border="0" width="100" height="0"
    cellspacing="0" cellpadding="0">
    <TR><TD></TD></TR>
  </TABLE>
</DIV>
<DIV id="mm5">
  <TABLE id="tablemm5" border="0" width="100" height="0"
    cellspacing="0" cellpadding="0">
    <TR><TD></TD></TR>
  </TABLE>
</DIV>
<DIV id="mm6">
  <TABLE id="tablemm6" border="0" width="82" height="60"
    cellspacing="0" cellpadding="0">
    <TR><TD id="mm6_1"><a href="mailto:Info@myline.com" name="mn6_1"
      onMouseOver="MenuOver_menu('mn6_1', 'mm6_1');
        window.status='General Info'; return true"
      onMouseOut="MenuOut_menu('mn6_1', 'mm6_1', 6);
        window.status=''; return true"
      class="submen2">&nbsp;General Info</a>
    </TD></TR>
    <TR><TD id="mm6_2"><a href="mailto:Sales@myline.com" name="mn6_2"
      onMouseOver="MenuOver_menu('mn6_2', 'mm6_2');
        window.status='Group Sales'; return true"
      onMouseOut="MenuOut_menu('mn6_2', 'mm6_2', 6);
        window.status=''; return true"
      class="submen2">&nbsp;Group Sales</a>
    </TD></TR>
    <TR><TD id="mm6_3"><a href="mailto:WebMaster@myline.com" name="mn6_3"
      onMouseOver="MenuOver_menu('mn6_3', 'mm6_3');
        window.status='Web Master'; return true"
      onMouseOut="MenuOut_menu('mn6_3', 'mm6_3', 6);
        window.status=''; return true"
      class="submen2">&nbsp;Web Master</a>
    </TD></TR>
  </TABLE>
</DIV>
```

```

</TABLE>

</DIV>

<CFSET LineMenu0='<TD width="6" align="center">
  <a href="Index.cfm" id="m1_0" class="men"
    onMouseOver=" mouse_over(' & chr(39) & 'm1_0' & chr(39) & ', 0, 0);
      window.status=' & chr(39) & 'Home' & chr(39) & '; return true"
    onMouseOut=" mouse_out(0);window.status=' & chr(39) & chr(39) & ';
      return true"
    onClick="m1_click(' & chr(39) & 'Index.cfm' & chr(39) & ');">&laquo;
  </a></TD>'>

<CFSET LineMenu1='<TD width="80" align="center">
  <a href="Cruises.cfm" id="m1_1" class="men"
    onMouseOver=" mouse_over(' & chr(39) & 'm1_1' & chr(39) & ', 1, 135);
      window.status=' & chr(39) & 'Cruises' & chr(39) & '; return true"
    onMouseOut=" mouse_out(1);window.status=' & chr(39) & chr(39) & ';
      return true"
    onClick="m1_click(' & chr(39) & 'Cruises.cfm' & chr(39) & ');">Cruises
  </a></TD>'>

<CFSET LineMenu2='<TD width="80" align="center">
  <a href="Schedules.cfm" id="m1_2" class="men"
    onMouseOver=" mouse_over(' & chr(39) & 'm1_2' & chr(39) & ', 2, 0);
      window.status=' & chr(39) & 'Schedules' & chr(39) & '; return true"
    onMouseOut=" mouse_out(2);window.status=' & chr(39) & chr(39) & ';
      return true" onClick="m1_click(' & chr(39) & 'Schedules.cfm' &
      chr(39) & ');">Schedules</a></TD>'>

<CFSET LineMenu3='<TD width="80" align="center">
  <a href="Guidance.cfm" id="m1_3" class="men"
    onMouseOver=" mouse_over(' & chr(39) & 'm1_3' & chr(39) & ', 3, 0);
      window.status=' & chr(39) & 'Guidance' & chr(39) & '; return true"
    onMouseOut=" mouse_out(3);window.status=' & chr(39) & chr(39) & ';
      return true" onClick="m1_click(' & chr(39) & 'Guidance.cfm' &
      chr(39) & ');">Guidance</a></TD>'>

<CFSET LineMenu4='<TD width="90" align="center">
  <a href="BuyTickets.cfm" id="m1_4" class="men"
    onMouseOver="mouse_over(' & chr(39) & 'm1_4' & chr(39) & ', 4, 0);
      window.status=' & chr(39) & 'Buy Tickets' & chr(39) & ';
      return true"
    onMouseOut=" mouse_out(4);window.status=' & chr(39) & chr(39) & ';

```

```

        return true"
    onClick="m1_click(' & chr(39) & 'BuyTickets.cfm' & chr(39) & ');">
    Buy Tickets</a></TD>'>
<CFSET LineMenu5='<TD width="80" align="center">
    <a href="CityMap.cfm" id="m1_5" class="men"
    onMouseOver=" mouse_over(' & chr(39) & 'm1_5' & chr(39) & ', 5, 0);
        window.status=' & chr(39) & 'City Map' & chr(39) & '; return true"
    onMouseOut=" mouse_out(5);window.status=' & chr(39) & chr(39) & ';
        return true" onClick="m1_click(' & chr(39) & 'CityMap.cfm' &
        chr(39) & ');">City Map</a></TD>'>
<CFSET LineMenu6='<TD width="80" align="center">
    <a href="ContactUs.cfm" id="m1_6" class="men"
    onMouseOver=" mouse_over(' & chr(39) & 'm1_6' & chr(39) & ', 6, 541);
        window.status=' & chr(39) & 'Contact Us' & chr(39) & '; return true"
    onMouseOut=" mouse_out(6);window.status=' & chr(39) & chr(39) & ';
        return true" onClick="m1_click(' & chr(39) & 'ContactUs.cfm' &
        chr(39) & ');">Contact Us</a></TD>'>
<CFSWITCH EXPRESSION=#CGI.SCRIPT_NAME#>
    <CFCASE VALUE="/MyLineCruises/Index.cfm">
        <CFSET LineMenu0='<TD width="6" align="center">
            <a class="men">&nbsp;&nbsp;&nbsp;</a></TD>'>
        </CFCASE>
        <CFCASE VALUE="/MyLineCruises/SightseeingCruises.cfm,/MyLineCruises/
        HarborLightsEvening.cfm,/MyLineCruises/SpecialEvents.cfm,/MyLineCruises/
        Cruises.cfm">
            <CFSET LineMenu1='<TD width="80" align="center">
                <a href="Cruises.cfm" id="m1_1" class="menact"
                onMouseOver=" mouse_over(' & chr(39) & 'm1_1' & chr(39) & ', 1, 135);
                    window.status=' & chr(39) & 'Cruises' & chr(39) & '; return true"
                onMouseOut=" mouse_out(1);window.status=' & chr(39) & chr(39) & ';
                    return true" onClick="m1_click(' & chr(39) & 'Cruises.cfm' &
                    chr(39) & ');">Cruises</a></TD>'>
            </CFCASE>
            <CFCASE VALUE="/MyLineCruises/Schedules.cfm">
                <CFSET LineMenu2='<TD width="80" align="center">
                    <a class="menact">Schedules</a></TD>'>
                </CFCASE>
                <CFCASE VALUE="/MyLineCruises/Guidance.cfm">

```

```

        <CFSET LineMenu3='<TD width="80" align="center">
        <a class="menact">Guidance</a></TD>'>
    </CFCASE>

    <CFCASE
VALUE="/MyLineCruises/BuyTickets.cfm,/MyLineCruises/SignUp.cfm,/
MyLineCruises/UpdateTra.cfm">
        <CFSET LineMenu4='<TD width="90" align="center"><a class="menact">Buy
Tickets</a></TD>'>
    </CFCASE>

    <CFCASE VALUE="/MyLineCruises/CityMap.cfm">
        <CFSET LineMenu5='<TD width="80" align="center">
        <a class="menact">City Map</a></TD>'>
    </CFCASE>

    <CFCASE VALUE="/MyLineCruises/ContactUs.cfm">
        <CFSET LineMenu6='<TD width="80" align="center">
        <a href="ContactUs.cfm" id="m1_6" class="menact"
onMouseOver=" mouse_over(' & chr(39) & 'm1_6' & chr(39) & ', 6, 541);
        window.status=' & chr(39) & 'Contact Us' & chr(39) & ';
        return true"
onMouseOut=" mouse_out(6);window.status=' & chr(39) & chr(39) & ';
        return true" onClick="m1_click(' & chr(39) & 'ContactUs.cfm' &
        chr(39) & ');">Contact Us</a></TD>'>
    </CFCASE>

    <CFDEFAULTCASE>
    </CFDEFAULTCASE>
</CFSWITCH>

<DIV id="tabm1">
    <TABLE width="516" cellspacing="0" cellpadding="0" border="0">
        <TR>
            <TD id="tdtabm1" nowrap>
                <TABLE border="0" width="516" cellspacing="0" cellpadding="0">
                    <TR>
                        <TD width="4" align="center" class="men">&nbsp;</TD>
                        <CFOUTPUT>#LineMenu0#</CFOUTPUT>
                        <CFOUTPUT>#LineMenu1#</CFOUTPUT>
                        <CFOUTPUT>#LineMenu2#</CFOUTPUT>
                        <CFOUTPUT>#LineMenu3#</CFOUTPUT>

```

```
<CFOUTPUT>#LineMenu4#</CFOUTPUT>
```

```
<CFOUTPUT>#LineMenu5#</CFOUTPUT>
```

```
<CFOUTPUT>#LineMenu6#</CFOUTPUT>
```

```
</TR>
```

```
</TABLE>
```

```
</TD>
```

```
</TR>
```

```
</TABLE>
```

```
</DIV>
```

Приложение 2



Коды основных файлов проекта "MyMail"

Данное приложение является дополнением *главы 16*, представляющей описание проекта "MyMail", и предназначено для демонстрации кода основных файлов данного проекта, в частности: Application.cfm, main.js, selectchecks.js, main.css, ie.css, login.cfm, signup.cfm, sendpassword.cfm, edituser.cfm, mailbox.cfm, receiving_mail.cfm, movedel_mail.cfm, emptydel.cfm, view_mail.cfm, properties_mail.cfm, newmail.cfm, addresses.cfm, addeditdel_addr.cfm, deletesel_addr.cfm, logout.cfm.

Листинг П2.1. Код конфигурационного файла Application.cfm

```
<CFAPPLICATION
    NAME="MyMail"
    APPLICATIONTIMEOUT="#createTimeSpan(0,1,0,0)#"
    SESSIONTIMEOUT="#createTimeSpan(0,0,20,0)#"
    SESSIONMANAGEMENT="Yes"
    CLIENTMANAGEMENT="Yes"
    SETCLIENTCOOKIES="Yes"
    CLIENTSTORAGE="registry">

<!-- Определение значения цвета фона (myBGColor) -->
<CFPARAM NAME="application.myBGColor" DEFAULT="FFFFFF">
<!-- Добавление стилей (Add Stylesheet) -->
<link rel="STYLESHEET" type="text/css" href="main.css">
<script type="text/javascript" language="JavaScript"
    src="main.js"></script>

<!-- Конфигурирование БД (Database configuration) -->
<CFPARAM NAME="application.dsn" DEFAULT="MyMail">
<!-- SMTP -->
<CFPARAM NAME="application.smtp" DEFAULT="192.168.3.68">
```



```

<!-- POP --->
<CFPARAM NAME="application.pop" DEFAULT="192.168.3.68">
<!-- Mail Server--->
<CFPARAM NAME="application.mailserver" DEFAULT="@MyMail">
<!-- Сообщения об ошибках (Message Errors) --->
<CFPARAM NAME="application.msgErrorLogin"
    DEFAULT="You have entered not the correct data!">
<CFPARAM NAME="application.msgError1"
    DEFAULT="Login should consist from more than 3 symbols">
<CFPARAM NAME="application.msgError2"
    DEFAULT="Password should consist from more than 4 symbols">
<CFPARAM NAME="application.msgError3"
    DEFAULT="Password should not consist only of numbers">
<CFPARAM NAME="application.msgError4"
    DEFAULT="Password and Confirm, should be identical">
<CFPARAM NAME="application.msgError5"
    DEFAULT="First Name, should not be empty">
<CFPARAM NAME="application.msgError6"
    DEFAULT="Last Name, should not be empty">
<!-- Параметры страниц (Parameters of pages) --->
<CFPARAM NAME="application.PageRows" DEFAULT="7">
<CFPARAM NAME="application.MaxLinePages" DEFAULT="12">

```

Листинг П2.2. Код файла main.js

```

var ns = (document.layers)? true:false
var ie = (document.all)? true:false

if (ie) {document.write('<link rel="stylesheet"
    type="text/css" href="ie.css" title="master">');}

function layerWrite(id,nestref,text) {
    if (ns) {
        var lyr = (nestref)?
eval('document.'+nestref+'.document.'+id+'.document') :
document.layers[id].document
        lyr.open()
        lyr.write(text)
    }
}

```

```
        lyr.close()
    }
    else if (ie) document.all[id].innerHTML = text
}
```

Листинг П2.3. Код файла selectchecks.js

```
function SelUnselInv(block) {
    if(ns) var idxlen =
document.layers[block].document.form2.elements.length;
    else var idxlen = document.form2.elements.length;
    return idxlen;
}

function SelectAll(block) {
    idxlen = SelUnselInv(block);
    for (i=0;i<idxlen;i++) {
        if(ns) document.layers[block].document.form2.elements[i].checked=true;
        else document.form2.elements(i).checked = true;
    }
}

function UnselectAll(block) {
    idxlen = SelUnselInv(block);
    for (i=0;i<idxlen;i++) {
        if(ns) document.layers[block].document.form2.elements[i].checked=false;
        else document.form2.elements(i).checked = false;
    }
}

function InvertChecks(block) {
    idxlen = SelUnselInv(block);
    for (i=0;i<idxlen;i++) {
        if(ns) {
            if
(document.layers[block].document.form2.elements[i].checked==true) {
                document.layers[block].document.form2.elements[i].checked=false;
            } else {
document.layers[block].document.form2.elements[i].checked=true;
            }
        }
    }
}
```

```
    } else {  
        if (document.form2.elements(i).checked==true) {  
            document.form2.elements(i).checked = false;  
        } else {  
            document.form2.elements(i).checked = true;  
        }  
    }  
}  
}
```

Листинг П2.4. Код файла main.css

```
A {color:#4B4558;}  
A:hover {color: #3366CC; text-decoration : underline;}  
  
.txtmenu {  
color : #4B4558;  
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;  
font-size : 12px;  
}  
  
.txtmenustat {  
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;  
font-size : 12px;  
padding : 3px;  
color : #4B4558;  
}  
  
.txtprompt {  
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;  
font-size : 11px;  
color : #4B4558;  
}  
  
.title {  
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;  
font-weight : bold;  
font-size : 17px;  
color : #4B4558;  
}
```

```
.txtForm1 {
padding : 0px;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
color : Black;
}

INPUT {
padding : 0px;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
}

TEXTAREA {
padding : 0px;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
}

.txtError {
padding : 0px;
font-family : Verdana, Geneva, Arial;
font-size : 11px;
color : Maroon;
}

.txt {
padding : 0px;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
color : Black;
}

.txtsearch {
padding : 0px;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
background : #e6e4dd;
text-decoration : none;
}

.txtlink {
padding : 0px;
```

```
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
color : #122C6D;
}

.txtact {
padding : 0px;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
color : #3366CC;
}

.note {
color : #B0B368;
}

#menu {position:absolute; left:10; top:30; width:550; height:20;
margin:0;z-index:1;}

#prompt {position:absolute; left:10; top:50; width:300;
height:30;margin:0;z-index:1;}

#basic {position:absolute; left:10; top:70; width:100%;
height:30;margin:0;z-index:1;}

#seladdr {position:absolute; left:10; top:-400; width:550; height:350;
z-index:2;}
```

Листинг П2.5. Код файла ie.css

```
.txtmenu {
color : #4B4558;
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
padding : 3px;
text-decoration : none;
font-weight : bold;
}

.txtmenustat {
font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 11px;
padding : 3px;
color : #4B4558;
}
```

```
.title {
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-weight : bold;
    font-size : 15px;
    color : #4B4558;
}

INPUT {
    padding : 0px;
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 10px;
}

TEXTAREA {
    padding : 0px;
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 10px;
}

.txtlink {
    padding : 0px;
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 11px;
    color : #122C6D;
    text-decoration : none;
}

.txtact {
    padding : 0px;
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 11px;
    color : #3366CC;
    text-decoration : none;
}
```

Листинг П2.6. Код страницы login.cfm

```
<HTML>
<HEAD>
    <TITLE>Login</TITLE>
</HEAD>
```

```

<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
  <script>
    function SendPassword() {
      if(ie) var EMAIL = document.form1.Email_User.value;
      else var EMAIL =
document.layers['basic'].document.form1.Email_User.value;
      if (EMAIL=="") {
        alert('Enter please Login(E-mail)');
      } else {
        location.href="sendpassword.cfm?Email="+EMAIL;
      }
    }
  </script>

  <CFIF isDefined("Session.user.id") AND Session.login eq "true">
    <!-- Пользователь уже существует
      (The user is already determined) --->
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=MailBox.cfm">
  <CFELSE>
    <CFIF isDefined("Form.Email_User") AND isDefined("Form.Pass_User")>
      <!-- Проверка (Verification) --->
      <CFQUERY NAME="VerificationUser" DATASOURCE="#Application.dsn#">
        SELECT Key_User, FirstN_User, LastN_User,
              Email_User, TotalLogins_User, Pass_User
        FROM UserMail
        WHERE Email_User = '#Form.Email_User#'
      </CFQUERY>
      <CFIF VerificationUser.RecordCount eq 1 AND Len(Form.Pass_User)
                                neq 0>
        <CFSET Password = Decrypt(VerificationUser.Pass_User,
                                VerificationUser.Key_User)>
        <CFIF Form.Pass_User eq Password>
          <CFSET Session.login = "true">
          <CFSET Session.user.id = #VerificationUser.Key_User#>
          <CFSET Session.user.name = #VerificationUser.FirstN_User# &
            " " & #VerificationUser.LastN_User#>
          <CFSET Session.user.email = #VerificationUser.Email_User#>
          <CFSET TotalLogins = #VerificationUser.TotalLogins_User# + 1>

```

```

<CFSET DateNow = #DateFormat(Now(), "dd/mm/yyyy")#& " " &
                        #TimeFormat(Now(), "HH:mm:ss")#>
<!--- Обновление записи (Update record) --->
<CFQUERY NAME="UpdateUser" DATASOURCE="#Application.dsn#">
    UPDATE UserMail SET LastIP_User = '#CGI.REMOTE_ADDR#',
                        LastBrowser_User = '#CGI.HTTP_USER_AGENT#',
                        LastLogin_User = '#DateNow#',
                        TotalLogins_User = #TotalLogins#
    WHERE Key_User = #Session.user.id#
</CFQUERY>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=mailbox.cfm">
<CFEXIT>
<CFELSE>
    <!--- Отказ в авторизации (Authenticate failure) --->
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm?error=1">
</CFIF>
<CFELSE>
    <!--- Отказ в авторизации (Authenticate failure) --->
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm?error=2">
</CFIF>
<CFELSE>
    <CFIF isDefined("Error")>
        <CFSET msgError="#Application.msgErrorLogin#">
    </CFIF>
<DIV id="menu">
    <TABLE width=440 border=0 cellpadding=0 cellspacing=0>
        <TR valign="Middle">
            <TD><SPAN class="txtmenustat">
                Log In&nbsp;&nbsp;&nbsp;&|&nbsp;&nbsp;&nbsp;&
                <A href="signup.cfm" target="_top" class="txtmenu"
                    onMouseOver="javascript:layerWrite('prompt',null, '<span
class=\&quot;txtprompt&quot; Sign up for reception of a Mail Box</span&quot;');
                    window.status='Sign up for reception of a Mail Box';
                    return true"
                    onMouseOut="javascript:layerWrite('prompt',null,'');
                    window.status=''; return true">Sign Up</A&nbsp;&nbsp;&nbsp;&|&nbsp;&nbsp;&nbsp;&
                <A href="help.cfm" target="_top" class="txtmenu"

```



```

        onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\txtprompt\>The auxiliary information</span>');
        window.status='The auxiliary information';
        return true"
        onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status='';
        return true">Help</A>
    </SPAN></TD>
</TR>
</TABLE>
</DIV>
<DIV id="prompt"><SPAN class="txtprompt"></SPAN></DIV>
<DIV id="basic">
<P class="title">Log In</P>
<FORM action="login.cfm" METHOD="POST" name="form1">
    <TABLE border="0" cellspacing="4" cellpadding="0">
        <CFIF isDefined("msgError")>
            <CFOUTPUT>
                <TR><TD colspan="2" align="center" class="txtError">
                    #msgError#</TD></TR>
            </CFOUTPUT>
        </CFIF>
        <TR><TD width="100" align="right" class="txtForm1">
            E-Mail (Login)</TD>
            <TD class="note"><input name="EMail_User" size="17"></TD>
        </TR>
        <TR><TD align="right" class="txtForm1">
            Password</TD><TD class="note">
                <input type="password" name="Pass_User" size="17"></TD></TR>
        <TR><TD align="right"><input type="submit" value="Send"></TD>
            <TD><input type="reset" value="Clear"></TD></TR>
        <TR><TD colspan="2" align="left" class="txtForm1">
            Forgot your password?<br><A href="javascript:SendPassword();"
                onMouseOver="window.status='Have it emailed to you';
                return true"
                onMouseOut="window.status=''; return true">
                Have it emailed to you.</A></TD></TR>
    </TABLE>

```

```

        </FORM>
    </DIV>
</CFIF>
</CFIF>
</BODY>
</HTML>

```

Листинг П2.7. Код страницы signup.cfm

```

<HTML>
<HEAD>
    <TITLE>Signup</TITLE>
</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
    <CFIF isDefined("Session.user.id") AND Session.login eq "true">
        <!-- Пользователь уже существует
             (The user is already determined) --->
        <META HTTP-EQUIV="Refresh" CONTENT="0;URL=mailbox.cfm">
    <CFELSE>
        <CFIF isDefined("Form.Email_User")>
            <CFSET Disuccessful = 1>
            <CFSET msgError = "Warning Error!">
            <CFIF Len(Form.Email_User) le 3>
                <CFSET msgError = msgError & "<br>" & #Application.msgError1#>
                <CFSET Disuccessful = 0>
            </CFIF>
            <CFIF Len(Form.Pass_User) le 4>
                <CFSET msgError = msgError & "<br>" & #Application.msgError2#>
                <CFSET Disuccessful = 0>
            </CFIF>
            <CFIF IsNumeric(Form.Pass_User)>
                <CFSET msgError = msgError & "<br>" & #Application.msgError3#>
                <CFSET Disuccessful = 0>
            </CFIF>
            <CFIF Form.Pass_User not equal Form.Confirm>
                <CFSET msgError = msgError & "<br>" & #Application.msgError4#>
                <CFSET Disuccessful = 0>
            </CFIF>

```

```

</CFIF>
<CFIF Len(Form.FirstN_User) le 0>
    <CFSET msgError = msgError & "<br>" & #Application.msgError5#>
    <CFSET DIsuccessful = 0>
</CFIF>
<CFIF Len(Form.LastN_User) le 0>
    <CFSET msgError = msgError & "<br>" & #Application.msgError6#>
    <CFSET DIsuccessful = 0>
</CFIF>
<CFIF YesNoFormat(DIsuccessful) equal "No">
<!-- Ошибка ввода (Error of input) -->
    <CFSET email = #Form.EMail_User#>
    <CFSET firstn = #Form.FirstN_User#>
    <CFSET lastn = #Form.LastN_User#>
<CFELSE>
    <!-- Проверка (Verification) -->
    <CFSET EMail_User = #Form.EMail_User# &
        #Application.mailserver#>
    <CFQUERY NAME="VerificationUser"
        DATASOURCE="#Application.dsn#">
        SELECT Key_User
        FROM UserMail
        WHERE EMail_User='#EMail_User#'
    </CFQUERY>
    <CFIF VerificationUser.RecordCount eq 1>
        <!-- Запись с этим адресом уже существует!
            (The record with this EMail address already exists!) -->
        <CFSET email = "">
        <CFSET firstn = #Form.FirstN_User#>
        <CFSET lastn = #Form.LastN_User#>
        <CFSET msgError = "The record with this EMail address
            already exists!">
    <CFELSE>
        <!-- Добавление записи (Add record) -->
        <CFSET DateNow = #DateFormat(Now(), "dd/mm/yyyy")#& " " &
            #TimeFormat(Now(), "HH:mm:ss")#>
        <CFQUERY NAME="AddUser" DATASOURCE="#Application.dsn#">

```

```

        INSERT INTO UserMail(Email_User, FirstN_User, LastN_User,
                               DateSignUp_User, LastLogin_User,
                               LastIP_User, LastBrowser_User,
                               TotalLogins_User)
        VALUES ('#Email_User#', '#Form.FirstN_User#',
                  '#Form.LastN_User#', '#DateNow#', '#DateNow#',
                  '#CGI.REMOTE_ADDR#', '#CGI.HTTP_USER_AGENT#', 1)
    </CFQUERY>
    <!-- Извлечение ключа (Extraction of a key) --->
    <CFQUERY NAME="UserExtKey" DATASOURCE="#Application.dsn#">
        SELECT Key_User
        FROM UserMail
        WHERE Email_User = '#Email_User#'
    </CFQUERY>
    <CFOUTPUT QUERY="UserExtKey">
        <CFSET Session.login = "true">
        <CFSET Session.user.id = #UserExtKey.Key_User#>
        <CFSET Session.user.name = #Form.FirstN_User# & " " &
                                   #Form.LastN_User#>
        <CFSET Session.user.email = #Email_User#>
    </CFOUTPUT>
    <CFSET Password = Encrypt(Form.Pass_User, Session.user.id)>
    <CFQUERY NAME="UpdatePass" DATASOURCE="#Application.dsn#">
        UPDATE UserMail SET Pass_User = '#Password#'
        WHERE Key_User = #Session.user.id#
    </CFQUERY>
    <META HTTP-EQUIV="Refresh"
CONTENT="0;URL=sendpassword.cfm?Email=<CFOUTPUT>#Session.user.email#
    </CFOUTPUT>&Type=SignUp">
    <CFEXIT>
</CFIF>
</CFIF>
<CFELSE>
    <CFSET email = "">
    <CFSET firstn = "">
    <CFSET lastn = "">
</CFIF>

```

```

<DIV id="menu">
  <TABLE width=440 border=0 cellpadding=0 cellspacing=0>
    <TR valign="middle">
      <TD><SPAN class="txtmenustat">
        <A href="login.cfm" target="_top" class="txtmenu"
          onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\txtprompt\>Data input for work with a Mail Box</span>');
          window.status='Data input for work with a Mail Box';
          return true"
          onMouseOut="javascript:layerWrite('prompt',null,'');
          window.status=''; return true">Log In</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
          Sign Up&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
        <A href="help.cfm" target="_top" class="txtmenu"
          onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\txtprompt\>The auxiliary information</span>');
          window.status='The auxiliary information';
          return true"
          onMouseOut="javascript:layerWrite('prompt',null,'');
          window.status=''; return true">Help</A>
      </SPAN></TD>
    </TR>
  </TABLE>
</DIV>

<DIV id="prompt"><SPAN class="txtprompt"></SPAN></DIV>
<DIV id="basic">
<P class="title">Sign Up</P>
<FORM action="signup.cfm" method="post" name="form1">
  <TABLE width="315" border="0" cellspacing="4" cellpadding="0">
    <CFIF isDefined("msgError")>
      <CFOUTPUT>
        <TR><TD colspan="2" align="center" class="txtError">
          #msgError#
        </TD></TR>
      </CFOUTPUT>
    </CFIF>
    <TR><TD align="right" class="txtForm1">
      E-Mail (Login)</TD><TD width="215" class="txtForm1">
        <input name="EMail_User" size="10"
          value="<CFOUTPUT>#email#</CFOUTPUT>"><CFOUTPUT>

```

```

        #Application.mailserver#</CFOUTPUT>&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    Password</TD><TD class="txtForm1">
        <input type="password" name="Pass_User"
            size="10">&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    Confirm</TD><TD class="txtForm1">
        <input type="password" name="Confirm"
            size="10">&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    First Name</TD><TD class="txtForm1">
        <input name="FirstN_User" size="17"
            value="<CFOUTPUT>#firstn#</CFOUTPUT>">&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
    Last Name</TD><TD class="txtForm1">
        <input name="LastN_User" size="17"
            value="<CFOUTPUT>#lastn#</CFOUTPUT>">&nbsp;*</TD></TR>
<TR><TD align="right">
        <input type="submit" value=" Add "></TD><TD>
        <input type="reset" value="Clear"></TD></TR>
</TABLE>
</FORM>
</DIV>
</CFIF>
</BODY>
</HTML>

```

Листинг П2.8. Код страницы sendpassword.cfm

```

<CFIF isDefined("URL.Email")>
    <CFQUERY NAME="Verification" DATASOURCE="#Application.dsn#">
        SELECT Pass_User, FirstN_User, Email_User
        FROM UserMail
        WHERE Email_User = '#URL.Email#'
    </CFQUERY>

```

```

<CFIF Verification.RecordCount eq 1>
  <CFSET PostMasterEMail = "Root" & #Application.mailserver#>
  <CFSET Password = Decrypt(Verification.Pass_User,
                           Verification.Key_User)>
  <!--- Отправка пароля (Send EMail Password) --->
  <CFMAIL TO="#URL.Email#"
    FROM="#PostMasterEMail#"
    SUBJECT="Password"
    SERVER="#Application.smtp#">
    Dear #Verification.FirstN_User#

    This message was sent at your request.
    For work with the project "#Application.ApplicationName#",
    you are registered E-Mail address(Login): #Verification.Email_User#
    with the password: #Password#

    Best wishes.
  </CFMAIL>
</CFIF>
<CFELSE>
  <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

<HTML>
  <BODY bgcolor="#<CFOUTPUT>#Application.myBGColor#</CFOUTPUT>">
    <CFIF isDefined("URL.Type")>
      <META HTTP-EQUIV="Refresh" CONTENT="0;URL=mailbox.cfm">
    <CFELSE>
      <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
    </CFIF>
  </BODY>
</HTML>

```

Листинг П2.9. Код страницы edituser.cfm

```

<HTML>
<HEAD>
  <TITLE>Edit User</TITLE>

```

```

</HEAD>
<BODY bgcolor="#Application.myBGColor">
  <CFIF isDefined("Session.user.name")>
    <CFIF isDefined("Form.Pass_User")>
      <CFSET Disuccessful = 1>
      <CFSET msgError = "Warning Error!">
      <CFIF Len(Form.Pass_User) le 4>
        <CFSET msgError = msgError & "<br>" & #Application.msgError2#>
        <CFSET Disuccessful = 0>
      </CFIF>
      <CFIF IsNumeric(Form.Pass_User)>
        <CFSET msgError = msgError & "<br>" & #Application.msgError3#>
        <CFSET Disuccessful = 0>
      </CFIF>
      <CFIF Form.Pass_User not equal Form.Confirm>
        <CFSET msgError = msgError & "<br>" & #Application.msgError4#>
        <CFSET Disuccessful = 0>
      </CFIF>
      <CFIF YesNoFormat(Disuccessful) equal "Yes">
        <!-- Обновить пароль (Update password) --->
        <CFSET Password = Encrypt(Form.Pass_User, Session.user.id)>
        <CFQUERY NAME="UpdateUser" DATASOURCE="#Application.dsn#">
          UPDATE UserMail
            SET Pass_User = '#Password#'
            WHERE EMail_User='#Session.user.email#'
        </CFQUERY>
        <META HTTP-EQUIV="Refresh">
CONTENT="#0;URL=sendpassword.cfm?EMail=<CFOUTPUT>#Session.user.email#
  </CFOUTPUT>&Type=EditUser">
  <CFEXIT>
</CFIF>
</CFIF>

<DIV id="menu">
  <TABLE width=440 border=0 cellpadding=0 cellspacing=0>
    <TR valign="Middle">
      <TD><span class="txtmenustat">
        <A href="mailbox.cfm" target="_top" class="txtmenu"

```



```

onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\"txtprompt\">Mail Box</span>');
    window.status='Mail Box';
    return true"
onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status=''; return true">Mail Box</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="newmail.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\"txtprompt\">New Mail</span>');
    window.status='New Mail'; return true"
onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status=''; return true">New Mail</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="addresses.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\"txtprompt\">Addresses</span>');
    window.status='Addresses'; return true"
onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status='';return true">Addresses</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="logout.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\"txtprompt\">Log Out</span>');
    window.status='Log Out'; return true"
onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status=''; return true">Log Out</A>
</SPAN></TD>
</TR>
</TABLE>
</DIV>

<DIV id="prompt"><span class="txtprompt"></span></DIV>

<DIV id="basic">
<p class="title">Edit User
<CFOUTPUT>#Session.user.name#</CFOUTPUT></p>

<FORM action="edituser.cfm" method="post" name="form1">
    <TABLE width="315" border="0" cellspacing="4" cellpadding="0">
        <CFIF isDefined("msgError")>
            <CFOUTPUT>

```

```

        <TR><TD colspan="2" align="center" class="txtError">
            #msgError#
        </TD></TR>
    </CFOUTPUT>
</CFIF>
<TR><TD align="right" class="txtForm1">
    E-Mail (Login)</TD><TD width="215" class="txtForm1">
        <CFOUTPUT>#Session.user.email#</CFOUTPUT></TD></TR>
<TR><TD align="right" class="txtForm1">
    Password</TD><TD class="txtForm1">
        <input type="password" name="Pass_User"
            size="10">&nbsp;   *</TD></TR>
<TR><TD align="right" class="txtForm1">
    Confirm</TD><TD class="txtForm1">
        <input type="password" name="Confirm"
            size="10">&nbsp;   *</TD></TR>
<TR><TD align="right">
        <input type="submit" value=" Set "></TD><TD>
        <input type="reset" value="Reset"></TD></TR>
</TABLE>
</FORM>
</DIV>
<CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>
</BODY>
</HTML>

```

Листинг П2.10. Код страницы mailbox.cfm

```

<CFIF isDefined("Session.user.name")>
    <CFQUERY NAME="User" DATASOURCE="#Application.dsn#">
        SELECT EMail_User, Pass_User
        FROM UserMail
        WHERE Key_User=#Session.user.id#
    </CFQUERY>
    <CFLOOP QUERY="User">

```

```
<CFSET Login = #Left(EMail_User, (FindOneOf("@", EMail_User)-1))#>
<CFSET Password = Decrypt(Pass_User, Session.user.id)>
</CFLOOP>
```

```
<CFQUERY NAME="CountNewMail" DATASOURCE="#Application.dsn#">
    SELECT Key_Box
    FROM MailBox
    WHERE Key_User=#Session.user.id#
    AND Key_TypeBox = 1 AND ReadState_Box = 0
</CFQUERY>
```

```
<CFIF not isDefined("URL.TypeBox")>
    <CFINCLUDE TEMPLATE="receiving_mail.cfm">
<CFELSE>
    <CFIF URL.TypeBox eq 1>
        <CFINCLUDE TEMPLATE="receiving_mail.cfm">
    <CFELSE>
        <CFSET CountNewMail = #CountNewMail.RecordCount#>
    </CFIF>
</CFIF>
```

```
<HTML>
<HEAD>
    <meta http-equiv="Expires" content="Tue, 01 Jan 1980 1:00:00 GMT">
    <meta http-equiv="Cache-Control" content="max-age=0">
    <TITLE>Mail Box</TITLE>
</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
    <script type="text/javascript" language="JavaScript"
        src="selectchecks.js"></script>
    <script>
        function MoveDeleteSelMail(block, type, what) {
            idxlen = SelUnselInv(block);
            var strCheck = "";
            for (i=0;i<idxlen;i++) {
                if (ns) {
                    if (document.layers[block].document.form2.elements[i].checked==
true) {
```

```

        if (strCheck=="") {
            strCheck =
document.layers[block].document.form2.elements[i].value;
        } else {
            strCheck = strCheck + "," +
            document.layers[block].document.form2.elements[i].value;
        }
    }
} else {
    if (document.form2.elements(i).checked==true) {
        if (strCheck=="") {
            strCheck = document.form2.elements(i).value;
        } else {
            strCheck = strCheck + "," +
            document.form2.elements(i).value;
        }
    }
}
}
if (strCheck == "") {
    alert("For moving or removal it is necessary
        to select even one mail!");
} else {
    eval("window.location.href =
'movedel_mail.cfm?Type="+type+"&What="+what+"&KeySel="+strCheck+"'");
}
}
}
</script>

<DIV id="menu">
<TABLE width=440 border=0 cellpadding=0 cellspacing=0>
<TR valign="Middle">
<TD><span class="txtmenustat">
Mail Box&nbsp;  |&nbsp;  
<A href="newmail.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,
'<span class=\'txtprompt\'>New Mail</span>');
window.status='New Mail'; return true"

```

```

        onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status='';return true">New Mail</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="addresses.cfm" target="_top" class="txtmenu"
    onMouseOver="javascript:layerWrite('prompt',null,
        '<span class=\'txtprompt\'>Addresses</span>');
        window.status='Addresses'; return true"
    onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status='';return true">Addresses</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="logout.cfm" target="_top"
    class="txtmenu"
    onMouseOver="javascript:layerWrite('prompt',null,
        '<span class=\'txtprompt\'>Log Out</span>');
        window.status='Log Out'; return true"
    onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status=''; return true">Log Out</A>
</span></TD>
</TR>
</TABLE>
</DIV>
<DIV id="prompt"><span class="txtprompt"></span></DIV>
<DIV id="basic">
    <CFIF isDefined("URL.TypeBox") >
        <CFSET TypeBox = #URL.TypeBox#>
    <CFELSE>
        <CFSET TypeBox = 1>
    </CFIF>
    <CFIF TypeBox eq 2 OR TypeBox eq 4>
        <CFSET ToFrom = "To">
    <CFELSE>
        <CFSET ToFrom = "From">
    </CFIF>
    <CFQUERY NAME="NameTypeBox" DATASOURCE="#Application.dsn#">
        SELECT Name_TypeBox
        FROM TypeBox
        WHERE Key_TypeBox=#TypeBox#
    </CFQUERY>
    <p class="title">Mail Box (
        <CFOUTPUT QUERY="NameTypeBox">#Name_TypeBox#</CFOUTPUT>

```

```

    <CFIF TextBox eq 1><CFOUTPUT>New - #CountNewMail#</CFOUTPUT>
  </CFIF>
)
<A href="edituser.cfm" class="txtlink">
<CFOUTPUT>#Session.user.name#</CFOUTPUT></A></p>
<CFIF isDefined("Order") AND isDefined("Trend")>
  <CFSET Order = #Order# & ' ' & #Trend# >
<CFELSE>
  <CFSET Order = 'Date_Box DESC'>
</CFIF>
<CFIF not isDefined("Search")>
  <CFSET Search = ''>
</CFIF>
<CFQUERY NAME="ListNameTextBox"
      DATASOURCE="#Application.dsn#">
  SELECT TextBox.Key_TypeBox, TextBox.Name_TypeBox
  FROM TextBox
  GROUP BY TextBox.Key_TypeBox, TextBox.Name_TypeBox
  ORDER BY TextBox.Key_TypeBox
</CFQUERY>
<CFQUERY NAME="CountTextBox" DATASOURCE="#Application.dsn#">
  SELECT TextBox.Key_TypeBox, Count (MailBox.Key_TypeBox)
  AS Count_TypeBox
  FROM TextBox, MailBox
  WHERE TextBox.Key_TypeBox = MailBox.Key_TypeBox
  AND ((MailBox.Key_User)=#Session.user.id#))
  GROUP BY TextBox.Key_TypeBox
  ORDER BY TextBox.Key_TypeBox
</CFQUERY>
<!--- Создание массива (Create Array) CountTextBoxArray --->
<CFSET CountTextBoxArray = ArrayNew(2)>
<CFLOOP QUERY="ListNameTextBox">
  <CFSET CountTextBoxArray[CurrentRow][1] =
    Name_TypeBox[CurrentRow]>
  <CFSET CountTextBoxArray[CurrentRow][2] = 0>
</CFLOOP>
<CFLOOP QUERY="CountTextBox">

```

```

<CFSET CountTypeBoxArray[Key_TypeBox[CurrentRow]][2] =
    Count_TypeBox[CurrentRow]>
</CFLOOP>

<CFQUERY NAME="ListMail" DATASOURCE="#Application.dsn#">
    SELECT MailBox.Key_Box, MailBox.To_Box, MailBox.From_Box,
        MailBox.Subject_Box, MailBox.Date_Box,
        MailBox.ReadState_Box
    FROM TypeBox INNER JOIN MailBox
    ON TypeBox.Key_TypeBox = MailBox.Key_TypeBox
    WHERE ((TypeBox.Key_TypeBox)=#TypeBox#)
        AND ((MailBox.Key_User)=#Session.user.id#))
        AND (#ToFrom#_Box Like '%#Search#%'
            OR Subject_Box Like '%#Search#%')
    ORDER BY #Order#;
</CFQUERY>

<FORM action="mailbox.cfm?TypeBox=<CFOUTPUT>#TypeBox#
    </CFOUTPUT>"
    method="post" name="form1">
    <input name="Search" size="20"
        value="<CFOUTPUT>#Search#</CFOUTPUT>">
    <input type="submit" value="Search">
    <CFIF TypeBox eq 3>
        <input type="Button" value="Empty Deleted"
            onClick="javascript:location.href='emptydel.cfm'">
    </CFIF>
</FORM>

<CFSET CurBlockPages = 1>
<CFIF isDefined("URL.CurPage")>
    <CFSET CurPage = #URL.CurPage#>
<CFELSE>
    <CFSET CurPage = 1>
</CFIF>

<CFIF isDefined("URL.StartPg") AND isDefined("URL.EndPg")>
    <CFSET StartPg = #URL.StartPg#>
    <CFSET EndPg = #URL.EndPg#>

```

```

<CFELSE>
  <CFSET StartPg = 1>
  <CFSET EndPg = Application.PageRows>
</CFIF>
<TABLE border="0" cellpadding="0" cellspacing="0">
  <TR><TD colspan="2" class="txt" align="right">
    Showing:
    <CFSET CountPages =
      Ceiling(ListMail.recordCount/Application.PageRows)>
  <CFLOOP INDEX = "Pages" FROM = "1" TO = "#CountPages#"
    <CFOUTPUT>
      <CFSET StartP =
        Pages*Application.PageRows-Application.PageRows+1>
      <CFSET EndP = Pages*Application.PageRows>
      <CFIF Pages eq CountPages>
        <CFSET EndP = ListMail.recordCount>
      </CFIF>
      <CFIF Pages eq CurPage>
        #StartP#-#EndP#
      <CFELSE>
        <A href="mailbox.cfm?CurPage
          =#Pages#&StartPg=#StartP#&EndPg
          =#EndP#&Search=#Search#&TypeBox=#TypeBox#"
          class="txtlink">#StartP#-#EndP#</A>
      </CFIF>
      <CFIF Pages neq CountPages>,</CFIF>
      <CFIF Pages eq Application.MaxLinePages*CurBlockPages>
        <CFSET CurBlockPages = CurBlockPages + 1>
        <br>
      </CFIF>
    </CFOUTPUT>
  </CFLOOP>
</TD></TR>
<TR><TD valign="top">
  <TABLE border="0" cellpadding="2" cellspacing="4">
    <TR><TD bgcolor="#cbccbd" height=1></TD></TR>
    <TR><TD class="txt">Folder</TD></TR>
    <TR><TD bgcolor="#cbccbd" height=1></TD></TR>

```


[illegible]

[illegible]

```

<CFELSE>txtsearch</CFIF>">
    <CFOUTPUT>#To_Box#</CFOUTPUT></TD>
<CFELSE>
    <TD class="<CFIF From_Box does not contain Search
OR Search eq ''>txt<CFELSE>txtsearch</CFIF>"><CFOUTPUT><CFIF Read-
State_Box eq 0><b></CFIF>#From_Box#<CFIF ReadState_Box eq
0></b></CFIF></CFOUTPUT></TD>
</CFIF>
    <TD class="<CFIF Subject_Box does not contain
Search OR Search eq ''>txt<CFELSE>txtsearch</CFIF>"><A
href="view_mail.cfm?Key=<CFOUTPUT>#Key_Box#</CFOUTPUT>&TypeBox=<CFOUTPUT>
#TypeBox#</CFOUTPUT>" class="txtlink"><CFOUTPUT><CFIF ReadState_Box eq
0><b></CFIF>#Subject_Box#<CFIF ReadState_Box eq
0></b></CFIF></CFOUTPUT></A></TD>
    <TD class="<CFIF Date_Box does not contain Search
OR Search eq ''>txt<CFELSE>txtsearch</CFIF>"><CFOUTPUT><CFIF Read-
State_Box eq 0><b></CFIF>#Date_Box#<CFIF ReadState_Box eq
0></b></CFIF></CFOUTPUT></TD>
    <TD class="txt" align="center">
        <CFIF TypeBox eq 1 OR TypeBox eq 3 OR TypeBox eq 5>
            <A
href="newmail.cfm?To=<CFOUTPUT>#From_Box#</CFOUTPUT>&Key=<CFOUTPUT>
#Key_Box#</CFOUTPUT>">
                </A>&nbsp;&nbsp;&nbsp;
            </CFIF>
            <CFIF TypeBox eq 4>
                <A href="newmail.cfm?To=<CFOUTPUT>#To_Box#</CFOUTPUT>
                    &Draft=yes&Key=<CFOUTPUT>#Key_Box#</CFOUTPUT>">
                    </A>&nbsp;&nbsp;&nbsp;
                </CFIF>
                <A href="view_mail.cfm?Key=<CFOUTPUT>#Key_Box#</CFOUTPUT>
                    &TypeBox=<CFOUTPUT>#TypeBox#</CFOUTPUT>">
                    </A>&nbsp;&nbsp;&nbsp;
                <A href="movedel_mail.cfm?Type=<CFOUTPUT>#statusDel#</CFOUTPUT>
                    &Key=<CFOUTPUT>#Key_Box#</CFOUTPUT>">
                    </A>&nbsp;&nbsp;&nbsp;
                <CFIF TypeBox eq 1>

```

```

<A href="movedel_mail.cfm?Type=Move&What=5&Key=<CFOUTPUT>
    #Key_Box#</CFOUTPUT>">
    </A>&nbsp;
</CFIF>
<CFIF TextBox eq 3 OR TextBox eq 5>
<A href="movedel_mail.cfm?Type=Move&What=1&Key=<CFOUTPUT>
    #Key_Box#</CFOUTPUT>">
    </A>&nbsp;
</CFIF>
</TD></TR>
</CFLOOP>
</CFIF>
<TR><TD colspan="5" bgcolor="#cbccbd"
    height=1></TD></TR>
<TR><TD colspan="4">
    <A href="javascript:SelectAll('basic');"
        class="txtlink">Select All
    </A>&nbsp;;
    <A href="javascript:UnselectAll('basic');"
        class="txtlink">Unselect All
    </A>&nbsp;;
    <A href="javascript:InvertChecks('basic');"
        class="txtlink">
        Invert Checks</A>
</TD>
<TD align="right">
    <A href="javascript:MoveDeleteSelMail('basic',
        '<CFOUTPUT>#statusDel#</CFOUTPUT>');">
    </A>&nbsp;
<CFIF TextBox eq 1>
<A href="javascript:MoveDeleteSelMail('basic',
    'Move', '5');">
</A>
</CFIF>

```

```

        <CFIF TypeBox eq 3 OR TypeBox eq 5>
        <A href="javascript:MoveDeleteSelMail('basic',
            'Move', '1');">
        </A>

    </CFIF>
</TD></TR>
</TABLE>
</TD></TR>
</TABLE>
</FORM>
</DIV>
</BODY>
</HTML>
<CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.11. Код страницы receiving_mail.cfm

```

<CFIF isDefined("Session.user.name")>
    <CFPOP SERVER= #Application.pop#
    USERNAME=#Login# PASSWORD=#Password#
    ATTACHMENTPATH="c:\temp"
    ACTION="GetAll" NAME="GetMail">

    <CFOUTPUT QUERY="GetMail">
        <CFSET Date_ = #DateFormat(ParseDateTime(Date), "dd/mm/yyyy")#& " " &
#TimeFormat(ParseDateTime(Date), "HH:mm:ss")#>
        <CFQUERY NAME="SaveMail" DATASOURCE="#Application.dsn#">
            <CFIF FindNoCase('Content-type: text/html', Header) gt 0>
                <CFSET conType = "HTML">
            <CFELSE>
                <CFSET conType = "">
            </CFIF>
            INSERT INTO MailBox(Key_User, Key_TypeBox, To_Box, From_Box,
                Cc_Box, Subject_Box, Body_Box, ReplyTo_Box,
                Type_Box, Date_Box, ReadState_Box, Header_Box,

```

```

        Attachments_Box, AttachmentFiles_Box)
VALUES (#Session.user.id#, 1, '#To#', '#From#',
        '#Cc#', '#Subject#', '#Body#', '#Replyto#',
        '#conType#', '#Date_#', 0, '#Header#',
        '#Attachments#', '#Attachmentfiles#')

</CFQUERY>

<CFPOP SERVER= #Application.pop#
        USERNAME=#Login# PASSWORD=#Password# MESSAGENUMBER=1
        ACTION="Delete" NAME="DelMail">

</CFOUTPUT>

<CFSET CountNewMail = #CountNewMail.RecordCount# +
#GetMail.RecordCount#>

<CFELSE>

    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">

</CFIF>

```

Листинг П2.12. Код страницы movedel_mail.cfm

```

<CFIF isDefined("Session.user.name")>

    <HTML>

    <HEAD>

        <TITLE>Move Delete Mail</TITLE>

    </HEAD>

    <BODY bgcolor="#<CFOUTPUT>#Application.myBGColor#</CFOUTPUT>">

        <CFIF isDefined("URL.Type")>

            <CFIF URL.Type eq "DeleteAbsolut">

                <CFIF isDefined("URL.Key")>

                    <CFQUERY NAME="DeleteMail" DATASOURCE="#Application.dsn#">
                        DELETE * FROM MailBox WHERE Key_Box = #URL.Key#
                    </CFQUERY>

                <CFELSE>

                    <CFIF isDefined("URL.KeySel")>

                        <CFQUERY NAME="DeleteMail" DATASOURCE="#Application.dsn#">
                            DELETE * FROM MailBox WHERE Key_Box IN (#URL.KeySel#)
                        </CFQUERY>

                    </CFIF>

                </CFIF>

            </CFIF>

```

```

<CFELSE>
    <CFIF URL.Type eq "Move" AND isDefined("URL.What")>
        <CFSET What = URL.What>
    <CFELSE>
        <CFSET What = 3>
    </CFIF>
    <CFIF isDefined("URL.Key")>
        <CFQUERY NAME="MoveMail" DATASOURCE="#Application.dsn#">
            UPDATE MailBox
                SET Key_TypeBox = #What#
                WHERE Key_Box = #URL.Key#
        </CFQUERY>
    <CFELSE>
        <CFIF isDefined("URL.KeySel")>
            <CFQUERY NAME="MoveMail" DATASOURCE="#Application.dsn#">
                UPDATE MailBox
                    SET Key_TypeBox = #What#
                    WHERE Key_Box IN (#URL.KeySel#)
            </CFQUERY>
        </CFIF>
    </CFIF>
</CFIF>
</CFIF>
</CFIF>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=mailbox.cfm">
<CFEXIT>
</BODY>
</HTML>
<CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.13. Код страницы emptydel.cfm

```

<CFIF isDefined("Session.user.name")>
    <HTML>
        <HEAD>
            <TITLE>Empty Delete Mail</TITLE>

```

```

</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
  <CFQUERY NAME="EmptyDeleteMail" DATASOURCE="#Application.dsn#">
    DELETE * FROM MailBox
      WHERE Key_User= #Session.user.id# AND Key_TypeBox = 3
  </CFQUERY>
  <META HTTP-EQUIV="Refresh" CONTENT="0;URL=mailbox.cfm">
</BODY>
</HTML>
<CFELSE>
  <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.14. Код страницы view_mail.cfm

```

<CFIF isDefined("Session.user.name")>
<CFIF isDefined("URL.Key")>
  <CFQUERY NAME="ViewMail" DATASOURCE="#Application.dsn#">
    SELECT *
      FROM MailBox
      WHERE Key_Box = #URL.Key#
  </CFQUERY>
<HTML>
  <HEAD>
    <TITLE>View Mail</TITLE>
  </HEAD>
  <BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
    <script>
      function Properties() {
eval("Variable=window.open('properties_mail.cfm?Key=<CFOUTPUT>#URL.Key#
</CFOUTPUT>','winviewmailprop','top=100, left=200, width=250,
height=320')");
      }
    </script>
    <DIV id="menu">
      <TABLE width=440 border=0 cellpadding=0 cellspacing=0>
        <TR valign="Middle">
          <TD><span class="txtmenustat">

```



```

Mail Box&nbsp;|&nbsp;
    <A href="newmail.cfm" target="_top" class="txtmenu"
        onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\'txtprompt\'>New Mail</span\'); window.status='New Mail'; return
true" onMouseOut="javascript:layerWrite('prompt',null,'');
window.status=''; return true">New Mail</A>&nbsp;|&nbsp;
    <A href="addresses.cfm" target="_top" class="txtmenu"
        onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\'txtprompt\'>Addresses</span\'); window.status='Addresses'; return
true" onMouseOut="javascript:layerWrite('prompt',null,'');
window.status=''; return true">Addresses</A>&nbsp;|&nbsp;
    <A href="logout.cfm" target="_top" class="txtmenu"
        onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\'txtprompt\'>Log Out</span\'); window.status='Log Out'; return
true" onMouseOut="javascript:layerWrite('prompt',null,'');
window.status=''; return true">Log Out</A>
</span></TD>
</TR>
</TABLE>
</DIV>

<DIV id="prompt"><span class="txtprompt"></span></DIV>

<DIV id="basic">
<CFQUERY NAME="TypeBox" DATASOURCE="#Application.dsn#">
    SELECT TypeBox.Name_TypeBox
    FROM TypeBox
    WHERE Key_TypeBox = #URL.TypeBox#
</CFQUERY>
<CFQUERY NAME="IMail" DATASOURCE="#Application.dsn#">
    SELECT Key_Box
    FROM MailBox
    WHERE Key_TypeBox = #URL.TypeBox#
        AND Key_User=#Session.user.id#
</CFQUERY>
<CFSET ListMail = ValueList(IMail.Key_Box)>

<p class="title">
    <A href="mailbox.cfm?TypeBox=<CFOUTPUT>#URL.TypeBox#</CFOUTPUT>">
        <CFOUTPUT>#TypeBox.Name_TypeBox#</CFOUTPUT></A> ( View Mail )

```

```

<A href="edituser.cfm" class="txtlink">
<CFOUTPUT>#Session.user.name#</CFOUTPUT></A>
</p>

```

```

<TABLE width="550" border="0" cellpadding="4" cellspacing="0">
<TR><TD align="right" class="txt" colspan="2">
<CFOUTPUT>
<CFIF ListFind(ListMail, URL.Key) gt 1>
  <A href="view_mail.cfm?Key=#ListFirst(ListMail)#
    &TypeBox=#URL.TypeBox#">
  </A>&nbsp;
  <A href="view_mail.cfm?Key=
    #ListGetAt(ListMail,ListFind(ListMail,URL.Key)-1)
    #&TypeBox=#URL.TypeBox#">
  </A>&nbsp;
</CFIF>
  #ListFind(ListMail, URL.Key)# of #ListLen(ListMail)#
  <CFIF ListFind(ListMail, URL.Key) lt ListLen(ListMail)>
    <A
href="view_mail.cfm?Key=#ListGetAt(ListMail,ListFind(ListMail,
URL.Key)+1)#&TypeBox=#URL.TypeBox#">
      </A>&nbsp;
    <A
href="view_mail.cfm?Key=#ListLast(ListMail)#&TypeBox=#URL.TypeBox#">
      </A>&nbsp;
    </CFIF>
  </CFOUTPUT>
</TD><TD class="txt">
<TR><TD align="right" class="txtForm1">From: </TD>
  <TD class="txt" width="450">
    <CFOUTPUT>#ViewMail.From_Box#</CFOUTPUT></TD></TR>
<TR><TD align="right" class="txtForm1">To: </TD>
  <TD class="txt" width="450">
    <CFOUTPUT>#ViewMail.To_Box#</CFOUTPUT></TD></TR>
<TR><TD align="right" class="txtForm1">Date: </TD>
  <TD class="txt">
    <CFOUTPUT>#ViewMail.Date_Box#</CFOUTPUT></TD></TR>
<TR><TD align="right" class="txtForm1">Subject: </TD>

```

[illegible]

```

        <CFIF ViewMail.Key_TypeBox eq 1>

        <A
href="movedel_mail.cfm?Type=Move&What=5&Key=<CFOUTPUT>#ViewMail.Key_Box#
</CFOUTPUT>">

        </A>&nbsp;
</CFIF>

        <CFIF ViewMail.Key_TypeBox eq 3 OR ViewMail.Key_TypeBox eq 5>

        <A
href="movedel_mail.cfm?Type=Move&What=1&Key=<CFOUTPUT>#ViewMail.Key_Box#
</CFOUTPUT>">

        </A>&nbsp;
</CFIF>
</TD></TR>
<TR><TD align="right" class="txtForm1" valign="top">Attachments:
        </TD>
        <TD>
                <CFLOOP INDEX="i" LIST=#ViewMail.AttachmentFiles_Box#
                        DELIMITERS=#Chr(9)#>
                        <CFOUTPUT>
                                <A href="file://#i#" class="txtlink">#i#</A><br>
                        </CFOUTPUT>
                </CFLOOP>
        </TD></TR>
<TR><TD align="right" class="txt" colspan="2">
        <FORM name="form1">
                <input type="Button" value="Properties"
                        onClick="javascript:Properties();">
        </FORM>
        </TD></TR>
</TABLE>
</DIV>

<CFIF ViewMail.ReadState_Box eq 0>
        <CFQUERY NAME="SetReadStateMail" DATASOURCE="#Application.dsn#">
                UPDATE MailBox
                SET ReadState_Box = 1
                WHERE Key_Box = #URL.Key#
        </CFQUERY>
</CFIF>

```

```

        </BODY>
    </HTML>
</CFIF>
<CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.15. Код страницы properties_mail.cfm

```

<CFIF isDefined("Session.user.name")>
    <CFIF isDefined("URL.Key")>
        <CFQUERY NAME="PropertiesMail" DATASOURCE="#Application.dsn#">
            SELECT ReplyTo_Box, Header_Box
            FROM MailBox
            WHERE Key_Box = #URL.Key#
        </CFQUERY>
        <HTML>
            <HEAD>
                <TITLE>Properties Mail</TITLE>
            </HEAD>
            <BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
                <TABLE width="100%" border="0" cellpadding="4" cellspacing="0">
                    <TD><TR class="txt">
                        <CFOUTPUT>#PropertiesMail.Header_Box#</CFOUTPUT></TD></TR>
                        <TR><TD class="txt">
                            <CFOUTPUT>#PropertiesMail.ReplyTo_Box#</CFOUTPUT></TD></TR>
                    </TABLE>

                </BODY>
            </HTML>
        </CFIF>
    <CFELSE>
        <script>
            window.close();
        </script>
    </CFIF>

```

Листинг П2.16. Код страницы newmail.cfm

```

<CFIF isDefined("Session.user.name")>
<CFIF isDefined("URL.SendType")>
<CFIF Form.SaveDraft eq "No" AND Form.UpdateDraft eq "No">
  <CFMAIL TO="#Form.To#"
    FROM="#Form.From#"
    CC="#Form.Cc#"
    BCC="#Form.Bcc#"
    SUBJECT="#Form.Subject#"
    SERVER="#Application.smtp#"
    TYPE="#Form.Type#"
    MIMEATTACH="#Form.Att1#">
    #Form.Body#
  <CFIF Form.Att2 neq "">
    <CFMAILPARAM FILE="#Form.Att2#">
    <CFSET Att2 = #CHR(9)# & #Form.Att2#>
  <CFELSE>
    <CFSET Att2 = "">
  </CFIF>
  <CFIF Form.Att3 neq "">
    <CFMAILPARAM FILE="#Form.Att3#">
    <CFSET Att3 = #CHR(9)# & #Form.Att3#>
  <CFELSE>
    <CFSET Att3 = "">
  </CFIF>
</CFMAIL>
<CFSET TypeBox = 2><!-- Исходящие (Outbox) -->
<CFELSE>
  <CFSET TypeBox = 4><!-- Черновики (Draft) -->
</CFIF>

<CFSET AttachmentFiles = Form.Att1 & Att2 & Att3>
  <CFSET DateNow = #DateFormat(Now(), "dd/mm/yyyy")#& " " &
#TimeFormat(Now(), "HH:mm:ss")#>
  <CFIF Form.UpdateDraft eq "Yes">
    <CFQUERY NAME="SaveMail" DATASOURCE="#Application.dsn#">
      UPDATE MailBox

```

```

    SET To_Box = '#Form.To#',
        Cc_Box = '#Form.Cc#',
        Bcc_Box = '#Form.Bcc#',
        Subject_Box = '#Form.Subject#',
        Body_Box = '#Form.Body#',
        Type_Box = '#Form.Type#',
        Date_Box = '#DateNow#',
        AttachmentFiles_Box = '#AttachmentFiles#'

    WHERE Key_Box = #Form.KeyBox#

</CFQUERY>

<CFELSE>

<CFQUERY NAME="SaveMail" DATASOURCE="#Application.dsn#">
    INSERT INTO MailBox(Key_User, Key_TypeBox, To_Box, From_Box,
                        Cc_Box, Bcc_Box, Subject_Box, Body_Box,
                        Type_Box, Date_Box, ReadState_Box,
                        AttachmentFiles_Box)
    VALUES(#Session.user.id#, #TypeBox#, '#Form.To#', '#Form.From#',
            '#Form.Cc#', '#Form.Bcc#', '#Form.Subject#', '#Form.Body#',
            '#Form.Type#', '#DateNow#', 1,
            '#AttachmentFiles#')

</CFQUERY>

</CFIF>

<META HTTP-EQUIV="Refresh" CONTENT="0;URL=mailbox.cfm">

<CFEXIT>

</CFIF>

<HTML>

<HEAD>

    <TITLE>New Mail</TITLE>

</HEAD>

<BODY on-
load="init();"bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">

    <script type="text/javascript" language="JavaScript"
        src="selectchecks.js"></script>

<script>
    function init() {
    }

    function Displacement(top, left) {
        if(ns) {

```

```
document.layers['seladdr'].top = top;
document.layers['basic'].left = left;
} else {
document.all['seladdr'].style.top = top;
document.all['basic'].style.left = left;
}
}

function SelAddr(field, block) {
if(ns) document.layers[block].document.form3.field.value = field;
else document.form3.field.value = field;
Displacement(70, -777);
}

function Cancel() {
Displacement(-400, 10);
}

function Done(block_, block) {
idxlen = SelUnselInv(block);
var strToCcBcc = "";
for (i=0;i<idxlen;i++) {
if (ns) {
if (document.layers[block].document.form2.elements[i].checked==
true) {
if (strToCcBcc=="") {
strToCcBcc = docu-
ment.layers[block].document.form2.elements[i].value;
} else {
strToCcBcc = strToCcBcc + "; " +
document.layers[block].document.form2.elements[i].value;
}
}
} else {
if (document.form2.elements(i).checked==true) {
if (strToCcBcc=="") {
strToCcBcc = document.form2.elements(i).value;
} else {
strToCcBcc = strToCcBcc + "; " +
document.form2.elements(i).value;
}
}
}
```



```

    }
    }
    }
    if (strToCcBcc == "") {
        alert("For application it is necessary to select even one address!");
    } else {
        if(ns) field = document.layers[block].document.form3.field.value;
        else field = document.form3.field.value;
        if(ns) eval("document.layers[block_].document.form1."+field+".value
= '' + strToCcBcc +'');
        else eval("document.form1."+field+".value = '' + strToCcBcc +'');
        Displacement(-400, 10);
    }
}

function DoneCur(block_, block, EMail) {
    if(ns) field = document.layers[block].document.form3.field.value;
    else field = document.form3.field.value;
    if(ns) eval("document.layers[block_].document.form1."+field+".value =
        '' + EMail +'');
    else eval("document.form1."+field+".value = '' + EMail +'');
    Displacement(-400, 10);
}

function IsForm1(fr) {
    if(fr.To.value.length==0){
        alert("Please, enter field - 'To'");
        return false;
    }
}

function IsForm3(fr, block_) {
    if(ns) Root = "document.layers[block_].document.form1"
    else Root = "document.form1"
    eval("fr.Subject.value = "+Root+".Subject.value");
    eval("fr.To.value = "+Root+".To.value");
    eval("fr.Cc.value = "+Root+".Cc.value");
    eval("fr.Bcc.value = "+Root+".Bcc.value");
    eval("fr.Body.value = "+Root+".Body.value");
    eval("fr.Type.value = "+Root+".Type.value");
}

```

```

    return true;
}

function TypeClick(val, block_) {
    if(ns) document.layers[block_].document.form1.Type.value = val;
    else    document.form1.Type.value = val;
}

function SDraft(block_) {
    if(ns) document.layers[block_].document.form1.SaveDraft.value =
        "Yes";

    else    document.form1.SaveDraft.value = "Yes";
}

function UDraft(block_) {
    if(ns) document.layers[block_].document.form1.UpdateDraft.value =
        "Yes";

    else    document.form1.UpdateDraft.value = "Yes";
}
</script>

```

```

<CFIF isDefined("Form.Search") OR isDefined("URL.CurPage")>
<CFOUTPUT>
<script>
    function init() {Displacement(70, -777);}
</script>
</CFOUTPUT>
</CFIF>

```

```

<DIV id="menu">
<TABLE width=440 border=0 cellpadding=0 cellspacing=0>
<TR valign="Middle">
<TD><span class="txtmenustat">
<A href="mailbox.cfm" target="_top" class="txtmenu"
    onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\txtprompt\''>Mail Box</span>');
    window.status='Mail Box'; return true"
    onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status=''; return true">Mail Box</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
    New Mail&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
    <A href="addresses.cfm" target="_top" class="txtmenu"

```

```

        onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\`txtprompt\`>Addresses</span>');
        window.status='Addresses'; return true"
        onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status=''; return true">Addresses</A>&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="logout.cfm" target="_top" class="txtmenu"
        onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\`txtprompt\`>Log Out</span>');
        window.status='Log Out'; return true"
        onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status=''; return true">Log Out</A>
    </span></TD>
</TR>
</TABLE>
</DIV>

<DIV id="prompt"><span class="txtprompt"></span></DIV>
<CFIF isDefined("To")>
    <CFSET to = #To#>
<CFELSE>
    <CFSET to = "">
</CFIF>
<CFSET keybox = 0>
<CFIF isDefined("Form.Subject")>
    <CFSET subject = #Form.Subject#>
    <CFSET field = #Form.field#>
    <CFSET cc = #Form.Cc#>
    <CFSET bcc = #Form.Bcc#>
    <CFSET body = #Form.Body#>
<CFIF #Form.Type# eq "">
    <CFSET type1 = "checked">
    <CFSET type2 = "">
    <CFSET typeVal = "">
<CFELSE>
    <CFSET type1 = "">
    <CFSET type2 = "checked">
    <CFSET typeVal = "HTML">
</CFIF>

```

```
<CFELSE>
<CFIF isDefined("URL.Key")>
  <CFQUERY NAME="EditMail" DATASOURCE="#Application.dsn#">
    SELECT *
      FROM MailBox
      WHERE Key_Box = #URL.Key#
  </CFQUERY>
  <CFSET keybox = #URL.Key#>
  <CFIF isDefined("URL.Draft")>
    <CFSET subject = #EditMail.Subject_Box#>
  <CFELSE>
    <CFSET subject = "Re: " & #EditMail.Subject_Box#>
  </CFIF>
  <CFSET field = "To">
  <CFSET cc = #EditMail.Cc_Box#>
  <CFSET bcc = #EditMail.Bcc_Box#>
  <CFSET body = #LTrim(EditMail.Body_Box)#>
  <CFIF #EditMail.Type_Box# eq "">
    <CFSET type1 = "checked">
    <CFSET type2 = "">
    <CFSET typeVal = "">
  <CFELSE>
    <CFSET type1 = "">
    <CFSET type2 = "checked">
    <CFSET typeVal = "HTML">
  </CFIF>
<CFELSE>
  <CFSET subject = "">
  <CFSET field = "To">
  <CFSET cc = "">
  <CFSET bcc = "">
  <CFSET body = "">
  <CFSET type1 = "checked">
  <CFSET type2 = "">
  <CFSET typeVal = "">
</CFIF>
</CFIF>
```

```

<DIV id="basic">
  <p class="title">New Mail
    <A href="edituser.cfm" class="txtlink">
      <CFOUTPUT>#Session.user.name#</CFOUTPUT></A></p>

  <FORM action="newmail.cfm?SendType=SendMail"
    method="post" name="form1" onSubmit="return IsForm1(this);">
    <TABLE width="400" border="0" cellspacing="4" cellpadding="0">
      <CFIF isDefined("msgError")>
        <CFOUTPUT>
          <TR><TD colspan="2" align="center" class="txtError">
            #msgError#
          </TD></TR>
        </CFOUTPUT>
      </CFIF>
      <TR>
        <TD align="right" class="txtForm1">From</TD>
        <TD class="txtForm1">
          <input name="From" size="20" readonly="Yes"
            value="<CFOUTPUT>#Session.user.email#</CFOUTPUT>">
        </TD></TR>
        <TR><TD align="right" class="txtForm1">
          <A href="javascript:SelAddr('To', 'seladdr');"
            class="txtlink">To</A></TD><TD class="txtForm1">
          <input name="To" size="56"
            value="<CFOUTPUT>#to#</CFOUTPUT>">&nbsp;&nbsp;&nbsp;&*</TD></TR>
        <TR><TD align="right" class="txtForm1">Subject</TD>
          <TD class="txtForm1"><input name="Subject" size="56"
            value="<CFOUTPUT>#subject#</CFOUTPUT>"></TD></TR>
        <TR><TD align="right" class="txtForm1">
          <A href="javascript:SelAddr('Cc', 'seladdr');"
            class="txtlink">Cc</A></TD><TD class="txtForm1">
          <input name="Cc" size="56"
            value="<CFOUTPUT>#cc#</CFOUTPUT>"></TD></TR>
        <TR><TD align="right" class="txtForm1">
          <A href="javascript:SelAddr('Bcc', 'seladdr');"
            class="txtlink">Bcc</A></TD><TD class="txtForm1">

```

```

        <input name="Bcc" size="56"
              value="<CFOUTPUT>#bcc#</CFOUTPUT>"></TD></TR>
<TR><TD class="txtForm1" colspan="2">
    <textarea name="Body" rows="8"
              cols="71"><CFOUTPUT>#body#</CFOUTPUT></textarea></TD></TR>
<TR><TD align="right" class="txtForm1">Type</TD>
    <TD class="txtForm1"><input type="radio" name="Type"
      onClick="TypeClick(' ', 'basic') "
      value=" "><CFOUTPUT>#type1#</CFOUTPUT>>plain text
    <input type="radio" name="Type"
      onclick="TypeClick('HTML', 'basic') "
      value="HTML"><CFOUTPUT>#type2#</CFOUTPUT>>html</TD></TR>
<TR><TD align="right" class="txtForm1">Attachment1</TD><TD>
    <input class="txtForm1" type="file" name="att1"
      size="36"></TD></TR>
<TR><TD align="right" class="txtForm1">Attachment2</TD><TD>
    <input class="txtForm1" type="file" name="att2"
      size="36"></TD></TR>
<TR><TD align="right" class="txtForm1">Attachment3</TD><TD>
    <input class="txtForm1" type="file" name="att3"
      size="36"></TD></TR>
<TR><TD align="right"><input type="submit"
      value="Send"></TD><TD><input type="reset" value="Clear">
<CFIF isDefined("URL.Draft")>
    <input type="submit" value="Update Draft"
      onClick="UDraft('basic') ">
<CFELSE>
    <input type="submit" value="Save Draft"
      onClick="SDraft('basic') ">
</CFIF>
<input type="Hidden" name="SaveDraft" value="No">
<input type="Hidden" name="UpdateDraft" value="No">
<input type="Hidden" name="KeyBox"
      value="<CFOUTPUT>#keybox#</CFOUTPUT>">
</TD></TR>
</TABLE>
</FORM>
</DIV>

```

```

<DIV id="seladdr">
  <p class="title">To/Cc/Bcc</p>
  <i class="txt">
    At change of the block of pages or sorting the
    parameters of the letter are not kept!</i>

  <CFIF isDefined("Order") AND isDefined("Trend")>
    <CFSET Order = #Order# & ' ' & #Trend# >
  <CFELSE>
    <CFSET Order = 'FirstN_Addr ASC'>
  </CFIF>

  <CFIF not isDefined("Search")>
    <CFSET Search = ''>
  </CFIF>

  <CFQUERY NAME="Addresses" DATASOURCE="#Application.dsn#">
    SELECT Key_Addr, FirstN_Addr, LastN_Addr,
           Company_Addr, EMail_Addr, Phone_Addr
    FROM Addresses
    WHERE Addresses.Key_User = #Session.user.id#
      AND (FirstN_Addr Like '%#Search#%' OR LastN_Addr Like '%#Search#%'
      OR Company_Addr Like '%#Search#%' OR EMail_Addr Like '%#Search#%')
    ORDER BY #Order#
  </CFQUERY>

  <FORM action="newmail.cfm" method="post" name="form3"
        onSubmit="IsForm3(this, 'basic')">
    <input name="Search" size="20"
          value="<CFOUTPUT>#Search#</CFOUTPUT>">
    <input type="submit" value="Search">
    <input type="Button" value="Done"
          onClick="javascript:Done('basic', 'seladdr');">
    <input type="Button" value="Cancel"
          onClick="javascript:Cancel();">
    <input type="Hidden" name="field">
    <input type="Hidden" name="Subject">
    <input type="Hidden" name="To">

```

```



```


[illegible]

```

<TR><TD colspan="6" bgcolor="#cbccbd" height=1></TD></TR>
<FORM method="post" name="form2">
  <CFLOOP QUERY="Addresses" STARTROW="#StartPg#" ENDROW="#EndPg#">
    <TR>
      <TD class="txt" align="center"><input type="checkbox"
        value="<CFOUTPUT>#Email_Addr#</CFOUTPUT>"
        name="<CFOUTPUT>#Key_Addr#</CFOUTPUT>"></TD>
      <TD class="<CFIF FirstN_Addr does not contain Search OR Search eq
''>txt<CFELSE>txtsearch</CFIF>"><CFOUTPUT>#FirstN_Addr#</CFOUTPUT></TD>
      <TD class="<CFIF LastN_Addr does not contain Search OR Search eq
''>txt<CFELSE>txtsearch</CFIF>"><CFOUTPUT>#LastN_Addr#</CFOUTPUT></TD>
      <TD class="<CFIF Company_Addr does not contain Search OR Search eq
''>txt<CFELSE>txtsearch</CFIF>"><CFOUTPUT>#Company_Addr#</CFOUTPUT></TD>
      <TD class="<CFIF Email_Addr does not contain Search OR Search eq
''>txt<CFELSE>txtsearch</CFIF>"><A href="javascript:DoneCur('basic',
'seladdr', '<CFOUTPUT>#Email_Addr#</CFOUTPUT>');" class="txtlink">
        <CFOUTPUT>#Email_Addr#</CFOUTPUT></A></TD>
      <TD class="txt"><CFOUTPUT>#Phone_Addr#</CFOUTPUT></TD>
    </TR>
  </CFLOOP>
  <TR><TD colspan="6" bgcolor="#cbccbd" height=1></TD></TR>
  <TR><TD colspan="5">
    <A href="javascript:SelectAll('seladdr');" class="txtlink">
      Select All</A>&nbsp;&nbsp;&nbsp;;
    <A href="javascript:UnselectAll('seladdr');" class="txtlink">
      Unselect All</A>&nbsp;&nbsp;&nbsp;;
    <A href="javascript:InvertChecks('seladdr');" class="txtlink">
      Invert Checks</A>
  </TD></TR>
</TABLE>
</FORM>
</DIV>
<script>
  if(ns) docu-
ment.layers['basic'].document.form1.Type.value="<CFOUTPUT>#typeVal#
</CFOUTPUT>";
  else document.form1.Type.value="<CFOUTPUT>#typeVal#</CFOUTPUT>";
  if(ns) docu-
ment.layers['seladdr'].document.form3.field.value="<CFOUTPUT>#field#
</CFOUTPUT>";

```

```

        else    document.form3.field.value="<CFOUTPUT>#field#</CFOUTPUT>";
    </script>

</BODY>
</HTML>
<CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.17. Код страницы addresses.cfm

```

<CFIF isDefined("Session.user.name")>
<HTML>
    <HEAD>
        <TITLE>Addresses</TITLE>
    </HEAD>
    <BODY bgcolor="<cfoutput>#Application.myBGColor#</cfoutput>">
        <script type="text/javascript" language="JavaScript"
            src="selectchecks.js"></script>
        <script>
            function SelCheck(block, delimiter, property) {
                idxlen = SelUnselInv(block);
                var strCheck = "";
                for (i=0;i<idxlen;i++) {
                    if(ns) {
                        if
                        (document.layers[block].document.form2.elements[i].checked==true) {
                            if (strCheck=="") {
                                eval("strCheck = docu-
ment.layers[block].document.form2.elements[i]."+property);
                            } else {
                                eval("strCheck = strCheck + delimiter + docu-
ment.layers[block].document.form2.elements[i]."+property);
                            }
                        }
                    } else {
                        if (document.form2.elements(i).checked==true) {
                            if (strCheck=="") {

```

```

        eval("strCheck = document.form2.elements(i)."+property);
    } else {
        eval("strCheck = strCheck + delimiter +
document.form2.elements(i)."+property);
    }
}
}
}
return strCheck;
}

function DeleteSelAddr(block) {
    strCheck = SelCheck(block, "," , "name");
    if (strCheck == "") {
        alert("For removal it is necessary to select even one address!");
    } else {
        eval("window.location.href = 'delete-
sel_addr.cfm?DelAddr="+strCheck+"'");
    }
}

function NewMailSelAddr(block) {
    strCheck = SelCheck(block, ";" , "value");
    if (strCheck == "") {
        alert("For creation of the new mail it is necessary
        to select even one address!");
    } else {
        eval("window.location.href = 'newmail.cfm?To="+strCheck+"'");
    }
}
}
</script>

<DIV id="menu">
    <TABLE width=440 border=0 cellpadding=0 cellspacing=0>
        <TR valign="Middle">
            <TD><span class="txtmenustat">
                <A href="mailbox.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,'<span
class=\`txtprompt\`>Mail Box</span>');
                window.status='Mail Box'; return true"

```

```

        onMouseOut="javascript:layerWrite('prompt',null,'');
        window.status=''; return true">Mail Box</A>&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
<A href="newmail.cfm" target="_top" class="txtmenu"
    onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\"txtprompt\"'>New Mail</span>');
    window.status='New Mail'; return true"
    onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status=''; return true">New Mail</A>&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
    Addresses&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;
    <A href="logout.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,'<span
class='\"txtprompt\"'>Log Out</span>');
    window.status='Log Out'; return true"
    onMouseOut="javascript:layerWrite('prompt',null,'');
    window.status=''; return true">Log Out</A>
</span></TD>
</TR>
</TABLE>
</DIV>

<DIV id="prompt"><span class="txtprompt"></span></DIV>

<DIV id="basic">
    <p class="title">Addresses
    <A href="edituser.cfm" class="txtlink">
    <CFOUTPUT>#Session.user.name#</CFOUTPUT></A></p>

    <CFIF isDefined("Order") AND isDefined("Trend")>
        <CFSET Order = #Order# & ' ' & #Trend# >
    <CFELSE>
        <CFSET Order = 'FirstN_Addr ASC'>
    </CFIF>
    <CFIF not isDefined("Search")>
        <CFSET Search = ''>
    </CFIF>
    <CFQUERY NAME="Addresses" DATASOURCE="#Application.dsn#">
        SELECT Key_Addr, FirstN_Addr, LastN_Addr,
            Company_Addr, EMail_Addr, Phone_Addr

```

```

FROM Addresses
WHERE Addresses.Key_User = #Session.user.id#
  AND (FirstN_Addr Like '%#Search#%' OR LastN_Addr Like '%#Search#%'
  OR Company_Addr Like '%#Search#%' OR EMail_Addr Like '%#Search#%')
ORDER BY #Order#
</CFQUERY>

<FORM action="addresses.cfm" method="post" name="form1">
  <input name="Search" size="20" value="#CFOUTPUT#>#Search#</CFOUTPUT#>">
  <input type="submit" value="Search">
  <input type="Button" value="New Contact"
    onClick="javascript:location.href='addeditdel_addr.cfm?Type=Add'">
  <input type="Button" value="New Mail for Selected"
    onClick="javascript:NewMailSelAddr('basic');">
</FORM>

<CFSET CurBlockPages = 1>
<CFIF isDefined("URL.CurPage")>
  <CFSET CurPage = #URL.CurPage#>
<CFELSE>
  <CFSET CurPage = 1>
</CFIF>

<CFIF isDefined("URL.StartPg") AND isDefined("URL.EndPg")>
  <CFSET StartPg = #URL.StartPg#>
  <CFSET EndPg = #URL.EndPg#>
<CFELSE>
  <CFSET StartPg = 1>
  <CFSET EndPg = Application.PageRows>
</CFIF>

<TABLE border="0" cellpadding="2" cellspacing="4">
  <TR><TD colspan="7" class="txt" align="right">
    Showing:
    <CFSET CountPages =
Ceiling(Addresses.recordCount/Application.PageRows)>
    <CFLOOP INDEX = "Pages" FROM = "1" TO = "#CountPages#">
    <CFOUTPUT>
    <CFSET StartP = Pages*Application.PageRows-Application.PageRows+1>
    <CFSET EndP = Pages*Application.PageRows>
    <CFIF Pages eq CountPages>

```

```

        <CFSET EndP = Addresses.recordCount>
    </CFIF>
    <CFIF Pages eq CurPage>
        #StartP#-#EndP#
    <CFELSE>
        <A
href="addresses.cfm?CurPage=#Pages#&StartPg=#StartP#&EndPg=#EndP#&Search=
#Search#" class="txtlink">#StartP#-#EndP#</A>
    </CFIF>
    <CFIF Pages neq CountPages>,</CFIF>
    <CFIF Pages eq Application.MaxLinePages*CurBlockPages>
        <CFSET CurBlockPages = CurBlockPages + 1>
        <br>
    </CFIF>
</CFOUTPUT>
</CFLOOP>
</TD></TR>
<TR><TD colspan="7" bgcolor="#cbccbd" height=1></TD></TR>
<TR><TD class="txt">To</TD>
    <TD class="txt">First
        <A href="addresses.cfm?Order=FirstN_Addr&Trend=ASC">
            </A>
        <A href="addresses.cfm?Order=FirstN_Addr&Trend=DESC">
            </A>&nbsp; &nbsp;  </TD>
    <TD class="txt">Last
        <A href="addresses.cfm?Order=LastN_Addr&Trend=ASC">
            </A>
        <A href="addresses.cfm?Order=LastN_Addr&Trend=DESC">
            </A>&nbsp; &nbsp;  </TD>
    <TD class="txt">Company
        <A href="addresses.cfm?Order=Company_Addr&Trend=ASC">
            </A>
        <A href="addresses.cfm?Order=Company_Addr&Trend=DESC">
            </A>&nbsp; &nbsp;  </TD>

```

```

<TD class="txt">Email
  <A href="addresses.cfm?Order=EEmail_Addr&Trend=ASC">
  </A>
  <A href="addresses.cfm?Order=EEmail_Addr&Trend=DESC">
  </A>&nbsp;&nbsp;&nbsp;</TD>
<TD class="txt">Phone</TD>
<TD class="txt" align="center">Controls</TD>
</TR>
<TR><TD colspan="7" bgcolor="#cbccbd" height=1></TD></TR>
<FORM method="post" name="form2">
  <CFIF Addresses.recordCount eq 0>
    <TR><TD class="txt" colspan="7" align="center">
      The address book is empty</TD></TR>
  <CFELSE>
  <CFLOOP QUERY="Addresses" STARTROW="#StartPg#" ENDROW="#EndPg#">
    <TR>
      <TD class="txt"><input type="checkbox"
        value="<CFOUTPUT>#Email_Addr#</CFOUTPUT>"
        name="<CFOUTPUT>#Key_Addr#</CFOUTPUT>"></TD>
      <TD class="<CFIF FirstN_Addr does not contain Search OR
        Search eq ''>txt<CFELSE>txtsearch</CFIF>">
        <CFOUTPUT>#FirstN_Addr#</CFOUTPUT></TD>
      <TD class="<CFIF LastN_Addr does not contain Search OR
        Search eq ''>txt<CFELSE>txtsearch</CFIF>">
        <CFOUTPUT>#LastN_Addr#</CFOUTPUT></TD>
      <TD class="<CFIF Company_Addr does not contain Search OR
        Search eq ''>txt<CFELSE>txtsearch</CFIF>">
        <CFOUTPUT>#Company_Addr#</CFOUTPUT></TD>
      <TD class="<CFIF EEmail_Addr does not contain Search OR
        Search eq ''>txt<CFELSE>txtsearch</CFIF>">
        <A href="newmail.cfm?To=<CFOUTPUT>#Email_Addr#</CFOUTPUT>"
          class="txtlink"><CFOUTPUT>#Email_Addr#</CFOUTPUT></A></TD>
      <TD class="txt"><CFOUTPUT>#Phone_Addr#</CFOUTPUT></TD>
      <TD class="txt" align="center">
        <A href="addeditdel_addr.cfm?Type=Edit&Key=<CFOUTPUT>
          #Key_Addr#</CFOUTPUT>">
        </A>&nbsp;&nbsp;&nbsp;

```



```

        <A href="addeditdel_addr.cfm?Type=Delete&Key=<CFOUTPUT>
            #Key_Addr#</CFOUTPUT>">
        </A></TD>
    </TR>
</CFLOOP>
</CFIF>
<TR><TD colspan="7" bgcolor="#cbccbd" height=1></TD></TR>
<TR><TD colspan="6">
    <A href="javascript:SelectAll('basic');"
        class="txtlink">Select All</A>&nbsp;&nbsp;&nbsp;;
    <A href="javascript:UnselectAll('basic');"
        class="txtlink">Unselect All</A>&nbsp;&nbsp;&nbsp;;
    <A href="javascript:InvertChecks('basic');"
        class="txtlink">Invert Checks</A>
</TD>
<TD align="right">
    <A href="javascript:DeleteSelAddr('basic');">
    </A>
</TD>
</TR>
</TABLE>
</FORM>
</DIV>

</BODY>
</HTML>
<CFELSE>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.18. Код страницы addeditdel_addr.cfm

```

<CFIF isDefined("Session.user.name")>
<HTML>
<HEAD>
    <TITLE><cfoutput>#Type#</cfoutput> Addresses</TITLE>
</HEAD>

```

```

<BODY bgcolor=" "><CFOUTPUT>#Application.myBGColor#</CFOUTPUT>">
<DIV id="menu">
<TABLE width=440 border=0 cellpadding=0 cellspacing=0>
<TR valign="Middle"><TD><span class="txtmenustat">
<A href="mailbox.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,
'<span class=\ 'txtprompt\ ' >Mail Box</span>');
window.status='Mail Box'; return true"
onMouseOut="javascript:layerWrite('prompt',null, '');
window.status=''; return true">Mail Box</A>&nbsp;  |&nbsp;  &nbsp; 
<A href="newmail.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,
'<span class=\ 'txtprompt\ ' >New Mail</span>');
window.status='New Mail'; return true"
onMouseOut="javascript:layerWrite('prompt',null, '');
window.status=''; return true">New Mail</A>&nbsp;  |&nbsp;  &nbsp; 
Addresses&nbsp;  |&nbsp;  &nbsp; 
<A href="logout.cfm" target="_top" class="txtmenu"
onMouseOver="javascript:layerWrite('prompt',null,
'<span class=\ 'txtprompt\ ' >Log Out</span>');
window.status='Log Out'; return true"
onMouseOut="javascript:layerWrite('prompt',null, '');
window.status=''; return true">Log Out</A>
</span></TD>
</TR>
</TABLE>
</DIV>
<DIV id="prompt"><span class="txtprompt"></span></DIV>
<DIV id="basic">
<p class="title"><CFOUTPUT>#Type#</CFOUTPUT>
<A href="addresses.cfm">Addresses</A>
<A href="edituser.cfm" class="txtlink">
<CFOUTPUT>#Session.user.name#</CFOUTPUT></A></p>
<CFIF isDefined("Form.EMail")>
<CFSET DIsuccessful = 1>
<CFSET msgError = "Warning Error!">
<CFIF Len(Form.EMail) le 5 and Form.EMail does not contain "@">
<CFSET msgError = msgError & "<br>" & #Application.msgError1#>

```

```

    <CFSET DIsuccessful = 0>
</CFIF>
<CFIF YesNoFormat(DIsuccessful) equal "No">
    <!--- Ошибки (Errors) --->
    <CFSET email = #Form.Email#>
    <CFSET firstn = #Form.FirstN#>
    <CFSET lastn = #Form.LastN#>
    <CFSET comp = #Form.Company#>
    <CFSET phone = #Form.Phone#>
<CFELSE>
    <!--- Проверка (Verification) --->
    <CFIF Type eq "Edit">
        <!--- Редактирование записи (Edit record) --->
        <CFQUERY NAME="EditAddr" DATASOURCE="#Application.dsn#">
            UPDATE Addresses
                SET Email_Addr = '#Form.Email#',
                    FirstN_Addr = '#Form.FirstN#',
                    LastN_Addr = '#Form.LastN#',
                    Company_Addr = '#Form.Company#',
                    Phone_Addr = '#Form.Phone#'
                WHERE Key_Addr = #Key#
        </CFQUERY>
    <CFELSE>
        <!--- Добавить запись (Add record) --->
        <CFQUERY NAME="AddAddr" DATASOURCE="#Application.dsn#">
            INSERT INTO Addresses(Key_User, Email_Addr, FirstN_Addr,
                                   LastN_Addr, Company_Addr, Phone_Addr)
            VALUES (#Session.user.id#, '#Form.Email#', '#Form.FirstN#',
                    '#Form.LastN#', '#Form.Company#', '#Form.Phone#')
        </CFQUERY>
    </CFIF>
    <META HTTP-EQUIV="Refresh" CONTENT="0;URL=addresses.cfm">
    <CFEXIT>
</CFIF>
<CFELSE>
    <CFSWITCH EXPRESSION=#Type#>
    <CFCASE VALUE="Edit">
        <CFQUERY NAME="SetAddr" DATASOURCE="#Application.dsn#">

```

```
SELECT *
FROM Addresses
WHERE Key_Addr = #URL.Key#
</CFQUERY>
<CFLOOP QUERY="SetAddr">
  <CFSET email = #SetAddr.EMail_Addr#>
  <CFSET firstn = #SetAddr.FirstN_Addr#>
  <CFSET lastn = #SetAddr.LastN_Addr#>
  <CFSET comp = #SetAddr.Company_Addr#>
  <CFSET phone = #SetAddr.Phone_Addr#>
</CFLOOP>
</CFCASE>
<CFCASE VALUE="Add">
  <CFSET email = "">
  <CFSET firstn = "">
  <CFSET lastn = "">
  <CFSET comp = "">
  <CFSET phone = "">
</CFCASE>
<CFCASE VALUE="Delete">
  <!-- Удалить запись (Delete record) -->
  <CFQUERY NAME="DeleteAddr" DATASOURCE="#Application.dsn#">
    DELETE * FROM Addresses WHERE Key_Addr = #URL.Key#
  </CFQUERY>
  <META HTTP-EQUIV="Refresh" CONTENT="0;URL=addresses.cfm">
  <CFEXIT>
</CFCASE>
</CFSWITCH>
</CFIF>
<FORM action="addeditdel_addr.cfm" method="post" name="form1">
  <TABLE width="300" border="0" cellpadding="2" cellspacing="0">
    <CFIF isDefined("msgError")>
      <CFOUTPUT>
        <TR><TD colspan="2" align="center" class="txtError">
          #msgError#</TD></TR>
      </CFOUTPUT>
    </CFIF>
    <TR><TD align="right" class="txtForm1">
```

```

First Name</TD><TD class="txtForm1">
<input name="FirstN" size="17"
      value="<CFOUTPUT>#firstn#</CFOUTPUT>"></TD></TR>
<TR><TD align="right" class="txtForm1">
Last Name</TD><TD class="txtForm1">
<input name="LastN" size="17"
      value="<CFOUTPUT>#lastn#</CFOUTPUT>"></TD></TR>
<TR><TD align="right" class="txtForm1">
Company</TD><TD width="215" class="txtForm1">
<input name="Company" size="25"
      value="<CFOUTPUT>#comp#</CFOUTPUT>"></TD></TR>
<TR><TD align="right" class="txtForm1">
E-Mail</TD><TD width="215" class="txtForm1">
<input name="EMail" size="25"
      value="<CFOUTPUT>#email#</CFOUTPUT>">&nbsp;*</TD></TR>
<TR><TD align="right" class="txtForm1">
Phone</TD><TD class="txtForm1">
<input name="Phone" size="12"
      value="<CFOUTPUT>#phone#</CFOUTPUT>"></TD></TR>
<TR><TD align="right">
<CFIF isDefined("URL.Key")>
  <input type="Hidden" name="Key"
        value="<CFOUTPUT>#URL.Key#</CFOUTPUT>">
</CFIF>
<input type="Hidden" name="Type"
      value="<CFOUTPUT>#Type#</CFOUTPUT>">
<input type="submit"
      value=" <CFOUTPUT>#Type#</CFOUTPUT> "></TD><TD>
  <input type="reset" value="Clear"></TD></TR>
</TABLE>
</FORM>
</DIV>

</BODY>
</HTML>
<CFELSE>
  <META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>

```

Листинг П2.19. Код страницы delete_sel_addr.cfm

```
<CFIF isDefined("Session.user.name")>
<HTML>
<HEAD>
<TITLE>Delete Select Addresses</TITLE>
</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
<CFIF isDefined("URL.DelAddr")>
<CFQUERY NAME="DeleteAddr" DATASOURCE="#Application.dsn#">
DELETE * FROM Addresses WHERE Key_Addr IN (#URL.DelAddr#)
</CFQUERY>
</CFIF>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=addresses.cfm">
</BODY>
</HTML>
<CFELSE>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
</CFIF>
```

Листинг П2.20. Код страницы logout.cfm

```
<CFSET Session.login = "false">
<CFSET Session.user.id = 0>
<CFSET Session.user.name = "">
<CFSET Session.user.email = "">
<HTML>
<HEAD>
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=login.cfm">
<TITLE>Logout</TITLE>
</HEAD>
<BODY bgcolor="#<cfoutput>#Application.myBGColor#</cfoutput>">
</BODY>
</HTML>
```

Код страницы help.cfm не рассматривается, т. к. ее содержание может быть абсолютно произвольным.



Функции CFML

Чтобы облегчить поиск и восприятие информации, данное приложение разделено на две части: в первой функции сгруппированы в алфавитном порядке с указанием условного кода, по которому можно найти описание той или иной функции во второй части данного приложения, где они сгруппированы в порядке принадлежности с кратким описанием и указанием соответствующего синтаксиса. Условный код представляет собой два числа, разделенных точкой, где первое число указывает на номер раздела принадлежности функции, а второе число представляет собой порядковый номер функции внутри раздела.

Например, номер функции `Abs`, равный 1.1, предполагает, что описание данной функции следует искать в первом разделе, а сама функция описана первой по порядку.

Заметим, что некоторые функции можно отнести сразу к нескольким разделам. К примеру, функцию `ListValueCount`, возвращающую количество вхождений заданного значения в списке, можно отнести как к символьным функциям, так и к функциям управления списками. Подобные функции мы будем относить только к одному наиболее подходящему по смыслу разделу применения, в данном случае, к разделу управления списками.

Алфавитный порядок

А

<code>Abs</code>	1.1	<code>ArrayPrepend</code>	9.11
<code>ACos</code>	1.2	<code>ArrayResize</code>	9.12
<code>ArrayAppend</code>	9.1	<code>ArraySet</code>	9.13
<code>ArrayAvg</code>	9.2	<code>ArraySort</code>	9.14
<code>ArrayClear</code>	9.3	<code>ArraySum</code>	9.15
<code>ArrayDeleteAt</code>	9.4	<code>ArraySwap</code>	9.16
<code>ArrayInsertAt</code>	9.5	<code>ArrayToList</code>	9.17

ArrayIsEmpty	9.6	Asc	2.1
ArrayLen	9.7	ASin	1.3
ArrayMax	9.8	Atn	1.4
ArrayMin	9.9	AuthenticatedContext	5.1
ArrayNew	9.10	AuthenticatedUser	5.2

B

BitAnd	1.5	BitOr	1.10
BitMaskClear	1.6	BitSHLN	1.11
BitMaskRead	1.7	BitSHRN	1.12
BitMaskSet	1.8	BitXor	1.13
BitNot	1.9		

C

Ceiling	1.14	CreateObject	13.1
Chr	2.2	CreateODBCDate	4.3
CJustify	2.3	CreateODBCDateTime	4.4
Compare	2.4	CreateODBCTime	4.5
CompareNoCase	2.5	CreateTime	4.6
Cos	1.15	CreateTimeSpan	4.7
CreateDate	4.1	CreateUUID	13.2
CreateDateTime	4.2		

D

DateAdd	4.8	DaysInMonth	4.17
DateCompare	4.9	DaysInYear	4.18
DateConvert	4.10	DE (Delay Evaluation)	13.3
DateDiff	4.11	DecimalFormat	12.2
DateFormat	12.1	DecrementValue	1.16
DatePart	4.12	Decrypt	2.6
Day	4.13	DeleteClientVariable	13.4
DayOfWeek	4.14	DirectoryExists	7.1

DayOfWeekAsString	4.15	DollarFormat	12.3
DayOfYear	4.16	Duplicate	13.5

E

Encrypt	2.7	Exp	1.17
Evaluate	13.6	ExpandPath	3.1

F

FileExists	7.2	FirstDayOfMonth	4.19
Find	2.8	Fix	1.18
FindNoCase	2.9	FormatBaseN	12.4
FindOneOf	2.10		

G

GetBaseTagData	13.7	GetHttpRequestTimeString	4.20
GetBaseTagList	13.8	GetMetricData	3.8
GetBaseTemplatePath	13.9	GetProfileString	3.9
GetClientVariablesList	13.10	GetTempDirectory	3.10
GetCurrentTemplatePath	3.2	GetTempFile	3.11
GetDirectoryFromPath	3.3	GetTemplatePath	3.12
GetException	3.4	GetTickCount	4.21
GetFileFromPath	3.5	GetTimeZoneInfo	4.22
GetFunctionList	3.6	GetToken	2.11
GetHttpRequestData	3.7		

H

Hash	6.1	HTMLCodeFormat	12.5
Hour	4.23	HTMLEditFormat	12.6

I

Iif	13.11	IsDate	7.9
IncrementValue	1.19	IsDebugMode	7.10

InputBaseN	1.20	IsDefined	7.11
Insert	2.12	IsLeapYear	7.12
Int	1.21	IsNumeric	7.13
IsArray	7.3	IsNumericDate	7.14
IsAuthenticated	7.4	IsProtected	7.15
IsAuthorized	7.5	IsQuery	7.16
IsBinary	7.6	IsSimpleValue	7.17
IsBoolean	7.7	IsStruct	7.18
IsCustomFunction	7.8	IsWDDX	7.19

J

JavaCast	2.13	JSStringFormat	2.14

L

LCase	2.15	ListToArray	10.18
Left	2.16	ListValueCount	10.19
Len	2.17	ListValueCountNoCase	10.20
ListAppend	10.1	LJustify	2.18
ListChangeDelims	10.2	Log	1.22
ListContains	10.3	Log10	1.23
ListContainsNoCase	10.4	LSCurrencyFormat	12.7
ListDeleteAt	10.5	LSDateFormat	12.8
ListFind	10.6	LSEuroCurrencyFormat	12.9
ListFindNoCase	10.7	LSIsCurrency	7.20
ListFirst	10.8	LSIsDate	7.21
ListGetAt	10.9	LSIsNumeric	7.22
ListInsertAt	10.10	LSNumberFormat	12.10
ListLast	10.11	LSParseCurrency	2.19
ListLen	10.12	LSParseDateTime	2.20
ListPrepend	10.13	LSParseEuroCurrency	2.21
ListQualify	10.14	LSParseNumber	2.22
ListRest	10.15	LSTimeFormat	12.11

ListSetAt	10.16	LTrim	2.23
ListSort	10.17		

M

Max	1.24	Minute	4.24
Mid	2.24	Month	4.25
Min	1.25	MonthAsString	4.26

N

Now	4.27	NumberFormat	12.12

P

ParagraphFormat	12.13	Pi	1.26
ParameterExists	7.23	PreserveSingleQuotes	13.12
ParseDateTime	4.28		

Q

Quarter	4.29	QueryNew	8.3
QueryAddColumn	8.1	QuerySetCell	8.4
QueryAddRow	8.2	QuotedValueList	10.21

R

Rand	1.27	ReplaceNoCase	2.31
Randomize	1.28	REReplace	2.32
RandRange	1.29	REReplaceNoCase	2.33
REFind	2.25	Reverse	2.34
REFindNoCase	2.26	Right	2.35
RemoveChars	2.27	RJustify	2.36
RepeatString	2.28	Round	1.30
Replace	2.29	RTrim	2.37
ReplaceList	2.30		

S

Second	4.30	StructDelete	11.5
SetLocale	13.13	StructFind	11.6
SetProfileString	3.13	StructFindKey	11.7
SetVariable	13.14	StructFindValue	11.8
Sgn	1.31	StructGet	11.9
Sin	1.32	StructInsert	11.10
SpanExcluding	2.38	StructIsEmpty	11.11
SpanIncluding	2.39	StructKeyArray	11.12
Sqr	1.33	StructKeyExists	11.13
StripCR	13.15	StructKeyList	11.14
StructAppend	11.1	StructNew	11.15
StructClear	11.2	StructSort	11.16
StructCopy	11.3	StructUpdate	11.17
StructCount	11.4		

T

Tan	1.34	ToBinary	6.3
TimeFormat	12.14	ToString	6.4
ToBase64	6.2	Trim	2.40

U

Ucase	2.41	URLEncodedFormat	6.6
URLDecode	6.5		

V

Val	2.42	ValueList	10.22

W

Week	4.31	WriteOutput	13.16

X

XMLFormat	6.7	

Y

Year	4.32	YesNoFormat	12.15

Категории функций

Будем использовать следующую классификацию функций:

- 1. Математические функции.
- 2. Символьные функции.
- 3. Системные функции.
- 4. Функции даты и времени.
- 5. Функции идентификации.
- 6. Функции преобразования.
- 7. Функции проверки.
- 8. Функции управления запросами.
- 9. Функции управления массивами.
- 10. Функции управления списками.
- 11. Функции управления структурами.
- 12. Функции форматирования.
- 13. Другие функции.

Математические функции

Функция	Описание	Код
Abc (number)	Возвращает абсолютное значение числа	1.1
ACos (number)	Возвращает косинус числа	1.2
ASin (number)	Возвращает синус числа	1.3
Atn (number)	Возвращает арктангенс числа	1.4
BitAnd (number1, number2)	Возвращает побитовое AND для двух длинных чисел	1.5
BitMaskClear (number, start, length)	Возвращает результат побитовой маски, который получается очисткой length битов, начиная с позиции start	1.6

(продолжение)

Функция	Описание	Код
BitMaskRead(number, start, length)	Возвращает результат побитовой маски, который получается чтением length битов из значения number, начиная с позиции start	1.7
BitMaskSet(number, mask, start, length)	Возвращает результат побитовой маски, заданной строкой mask, применимой к значению number длиной length битов, начиная с позиции start	1.8
BitNot(number)	Возвращает результат операции побитового NOT указанного числа	1.9
BitOr(number1, number2)	Возвращает побитовое OR для двух длинных чисел	1.10
BitSHLN(number, count)	Возвращает число, сдвинутое на заданное количество битов влево	1.11
BitSHRN(number, count)	Возвращает число, сдвинутое на заданное количество битов вправо	1.12
BitXor(number1, number2)	Возвращает XOR для двух длинных чисел	1.13
Ceiling(number)	Возвращает ближайшее целое большее, чем заданное число	1.14
Cos(number)	Возвращает косинус угла, заданного в радианах	1.15
DecrementValue(number)	Возвращает целую часть числа, уменьшенную на единицу	1.16
Exp(number)	Возвращает экспоненту числа	1.17
Fix(number)	Возвращает ближайшее целое меньшее, чем заданное число	1.18
IncrementValue(number)	Возвращает целую часть числа, увеличенную на единицу	1.19
InputBaseN(string, radix)	Возвращает число, полученное из строки конвертированием в число по заданному основанию	1.20
Int(number)	Возвращает ближайшее целое меньшее заданного числа	1.21
Log(number)	Возвращает двоичный логарифм числа	1.22
Log10(number)	Возвращает десятичный логарифм числа	1.23
Max(number1, number2)	Возвращает максимальное значение	1.24

(окончание)

Функция	Описание	Код
<code>Min(number1, number2)</code>	Возвращает минимальное значение	1.25
<code>Pi()</code>	Возвращает значение числа π	1.26
<code>Rand()</code>	Возвращает случайное десятичное число в диапазоне [0; 1)	1.27
<code>Randomize(number)</code>	Запускает генератор случайных чисел	1.28
<code>RandRange(number1, number2)</code>	Возвращает случайное десятичное число в диапазоне указанных чисел	1.29
<code>Round(number)</code>	Возвращает округленное число	1.30
<code>Sgn(number)</code>	Возвращает знак числа	1.31
<code>Sin(number)</code>	Возвращает синус числа	1.32
<code>Sqr(number)</code>	Возвращает квадратный корень числа	1.33
<code>Tan(number)</code>	Возвращает тангенс числа	1.34

Символьные функции

Функция	Описание	Код
<code>Asc(string)</code>	Возвращает значение восьмибитового расширенного ASCII-кода первого символа строки	2.1
<code>Chr(value)</code>	Возвращает символ восьмибитового расширенного ASCII-кода	2.2
<code>CJustify(string, length)</code>	Выравнивает строку по центру в поле заданной длины	2.3
<code>Compare(string1, string2)</code>	Выполняет сравнение двух строк	2.4
<code>CompareNoCase(string1, string2)</code>	Выполняет сравнение двух строк без учета регистра	2.5
<code>Decrypt(encrypted_string, seed)</code>	Расшифровывает строку по указанному ключу	2.6
<code>Encrypt(string, seed)</code>	Зашифровывает строку по указанному ключу	2.7
<code>Find(substring, string [, start])</code>	Возвращает индекс первого вхождения подстроки в строку, начиная с заданной позиции	2.8

(продолжение)

Функция	Описание	Код
<code>FindNoCase(substring, string [, start])</code>	Возвращает индекс первого вхождения подстроки в строку, начиная с заданной позиции без учета регистра	2.9
<code>FindOneOf(set, string [, start])</code>	Возвращает индекс первого вхождения в строку любого символа из заданного набора, начиная с заданной позиции	2.10
<code>GetToken(string, index [, delimiters])</code>	Возвращает элемент строки из заданного перечисления с учетом указанного разделителя	2.11
<code>Insert(substring, string, position)</code>	Вставляет подстроку в строку с указанной позиции	2.12
<code>JavaCast(type, variable)</code>	Передает переменную ColdFusion с указанным типом в качестве аргумента перезагружаемому методу объекта Java	2.13
<code>JSStringFormat(string)</code>	Возвращает строку, которую безопасно использовать вместе с кодом JavaScript	2.14
<code>LCase(string)</code>	Возвращает строку, преобразованную в нижний регистр	2.15
<code>Left(string, count)</code>	Возвращает указанное число символов с левого края строки	2.16
<code>Len(string or binary object)</code>	Возвращает длину строки	2.17
<code>Ljustify(string, length)</code>	Выравнивает строку по левому краю в поле заданной длины	2.18
<code>LSParseCurrency(string)</code>	Преобразует строку, представляющую валюту, в строку в виде числа	2.19
<code>LSParseDateTime(date/time)</code>	Преобразует строку с датой и временем в данные формата "дата/время"	2.20
<code>LSParseEuroCurrency(currency)</code>	Преобразует строку, представляющую евровалюту в строку в виде числа	2.21
<code>LSParseNumber(string)</code>	Преобразует строку в число	2.22
<code>LTrim(string)</code>	Ликвидирует пробелы слева в указанной строке	2.23
<code>Mid(string, start, count)</code>	Возвращает <code>count</code> символов из указанной строки, начиная с позиции, определяемой параметром <code>start</code>	2.24

(продолжение)

Функция	Описание	Код
<code>REFind(reg_expression, string [, start] [, returnsubexpressions])</code>	Возвращает позицию первого вхождения регулярного выражения в строке, начиная с заданной позиции	2.25
<code>REFindNoCase(reg_expression, string [, start] [, returnsubexpressions])</code>	Возвращает позицию первого вхождения регулярного выражения в строке без учета регистра, начиная с заданной позиции	2.26
<code>RemoveChars(string, start, count)</code>	Возвращает строку, в которой удалено указанное число символов, начиная с заданной позиции	2.27
<code>RepeatString(string, count)</code>	Возвращает строку, преобразованную из заданной строки методом повторения указанное количество раз	2.28
<code>Replace(string, substring1, substring2 [, scope])</code>	Заменяет в строке один фрагмент текста на другой	2.29
<code>ReplaceList(string, list1, list2)</code>	Заменяет в строке один список на другой	2.30
<code>ReplaceNoCase(string, substring1, substring2 [, scope])</code>	Заменяет в строке один фрагмент текста на другой без учета регистра	2.31
<code>REReplace(string, reg_expression, substring [, scope])</code>	Производит замену на основе регулярного выражения	2.32
<code>REReplaceNoCase(string, reg_expression, substring [, scope])</code>	Производит замену на основе регулярного выражения без учета регистра	2.33
<code>Reverse(string)</code>	Возвращает строку с обратным порядком символов	2.34
<code>Right(string, count)</code>	Возвращает указанное число символов с правого края строки	2.35
<code>RJustify(string, length)</code>	Выравнивает строку по правому краю в поле заданной длины	2.36
<code>RTrim(string)</code>	Ликвидирует пробелы справа в указанной строке	2.37
<code>SpanExcluding(string, set)</code>	Возвращает строку, в которой исключена часть, начиная с указанного набора символов	2.38

(окончание)

Функция	Описание	Код
<code>SpanIncluding(string, set)</code>	Возвращает строку, включая только часть, указанную в наборе символов	2.39
<code>Trim(string)</code>	Ликвидирует пробелы указанной строки с обеих сторон	2.40
<code>UCase(string)</code>	Преобразует символы в строке к верхнему регистру	2.41
<code>Val(string)</code>	Возвращает числовое значение из указанной строки	2.42

Системные функции

Функция	Описание	Код
<code>ExpandPath(relative_path)</code>	Возвращает путь, полученный методом добавления параметра к базовому пути страницы	3.1
<code>GetCurrentTemplatePath()</code>	Возвращает путь текущей страницы	3.2
<code>GetDirectoryFromPath(path)</code>	Извлекает имя каталога из пути	3.3
<code>GetException(object)</code>	Позволяет выявлять исключение относительно Java-объекта и используется в сочетании с тегами <code><CFTRY></code> и <code><CFCATCH></code>	3.4
<code>GetFileFromPath(path)</code>	Возвращает имя файла из полного пути	3.5
<code>GetFunctionList()</code>	Возвращает список функций	3.6
<code>GetHttpRequestData()</code>	Создает список HTTP-заголовков	3.7
<code>GetMetricData(mode)</code>	Возвращает метрические параметры	3.8
<code>GetProfileString(iniPath, section, entry)</code>	Возвращает строку профиля	3.9
<code>GetTempDirectory()</code>	Возвращает имя временного каталога	3.10
<code>GetTempFile(dir, prefix)</code>	Создает и возвращает имя временного файла из временного каталога	3.11
<code>GetTemplatePath()</code>	Возвращает полный базовый путь страницы	3.12
<code>SetProfileString(iniPath, section, entry, value)</code>	Устанавливает значение профиля в файле инициализации	3.13

Функции даты и времени

Функция	Описание	Код
CreateDate(year, month, day)	Создает объект формата "дата"	4.1
CreateDateTime(year, month, day, hour, minute, second)	Создает объект формата "дата/ время"	4.2
CreateODBCDate(date)	Создает дату в формате ODBC (d '2001-10-18')	4.3
CreateODBCDateTime(date)	Создает дату и время в формате ODBC	4.4
CreateODBCTime(date)	Создает время в формате ODBC	4.5
CreateTime(hour, minute, second)	Создает объект формата "время"	4.6
CreateTimeSpan(days, hours, minutes, seconds)	Создает объект формата "время" для добавления и вычитания других аналогичных объектов	4.7
DateAdd(datepart, number, date)	Добавляет число к дате	4.8
DateCompare(date1, date2 [, datePart])	Выполняет сравнение дат	4.9
DateConvert(conversion-type, date)	Выполняет преобразование даты	4.10
DateDiff(datepart, date1, date2)	Рассчитывает интервал между датами	4.11
DatePart(datepart, date)	Возвращает часть даты как целое число	4.12
Day(date)	Возвращает день месяца от 1 до 31	4.13
DayOfWeek(date)	Возвращает день недели от 1 до 7	4.14
DayOfWeekAsString(day_of_week)	Возвращает наименование дня недели, представленного числом	4.15
DayOfYear(date)	Возвращает порядковый номер дня в году от 1 до 366	4.16
DaysInMonth(date)	Возвращает число дней в месяце	4.17
DaysInYear(date)	Возвращает число дней в году	4.18
FirstDayOfMonth(date)	Возвращает порядковый номер первого дня в году относительно указанной даты	4.19

(окончание)

Функция	Описание	Код
GetHttpTimeString (date_time_object)	Возвращает преобразованную дату и время согласно стандарту HTTP в виде "Thu, 18 Oct 2001 17:59:55 GMT"	4.20
GetTickCount()	Возвращает значение таймера в миллисекундах, позволяющее отслеживать время выполнения фрагментов кода	4.21
GetTimeZoneInfo()	Возвращает код временной зоны	4.22
Hour(date)	Возвращает номер часа из указанной даты	4.23
Minute(date)	Возвращает минуту из указанной даты	4.24
Month(date)	Возвращает месяц из указанной даты	4.25
MonthAsString(month_number)	Возвращает название месяца, представленного числом	4.26
Now()	Возвращает текущую дату	4.27
ParseDateTime(date/time [, pop-conversion])	Возвращает значение даты и времени, полученное из строки	4.28
Quarter(date)	Возвращает номер квартала из указанной даты	4.29
Second(date)	Возвращает секунды из указанной даты	4.30
Week(date)	Возвращает номер недели из указанной даты	4.31
Year(date)	Возвращает год из указанной даты	4.32

Функции идентификации

Функция	Описание	Код
AuthenticatedContext()	Возвращает имя идентифицированного содержания	5.1
AuthenticatedUser()	Возвращает имя идентифицированного пользователя	5.2

Функции преобразования

Функция	Описание	Код
<code>Hash(string)</code>	Преобразует текст в 32-битный код по одностороннему алгоритму MD5	6.1
<code>ToBase64(string or binary_object)</code>	Преобразует формат строки в кодировку Base64	6.2
<code>ToBinary(string_in_Base64 or binary_value)</code>	Преобразует данные в двоичный формат	6.3
<code>ToString(any_value)</code>	Преобразует данные в строку	6.4
<code>URLDecode(urlEncodedString)</code>	Выполняет обратное преобразование для URL-кодированной строки	6.5
<code>URLEncodedFormat(string)</code>	Выполняет кодирование данных для включения в строку URL	6.6
<code>XMLFormat(string)</code>	Преобразует строку в XML-формат	6.7

Функции проверки

Функция	Описание	Код
<code>DirectoryExists(absolute_path)</code>	Возвращает <code>True</code> , если каталог существует	7.1
<code>FileExists(absolute_path)</code>	Возвращает <code>True</code> , если файл существует	7.2
<code>IsArray(value [, number])</code>	Возвращает <code>True</code> , если значение является массивом, где <code>number</code> — число, позволяющее уточнить проверку на соответствие количества элементов массива	7.3
<code>IsAuthenticated([security-context-name])</code>	Возвращает <code>True</code> , если пользователь был идентифицирован в корректном окружении	7.4
<code>IsAuthorized(resourceType, resourceName [, action])</code>	Возвращает <code>True</code> , если пользователь авторизован для выполнения действия над ресурсом ColdFusion	7.5
<code>IsBinary(value)</code>	Возвращает <code>True</code> , если значение является двоичным целым	7.6
<code>IsBoolean(value)</code>	Возвращает <code>True</code> , если значение может быть преобразовано в логическое значение	7.7

(продолжение)

Функция	Описание	Код
<code>IsCustomFunction(name)</code>	Возвращает <code>True</code> , если существует пользовательская функция с указанным именем	7.8
<code>IsDate(string)</code>	Возвращает <code>True</code> , если указанную строку можно преобразовать, используя одну из функций по работе с датами	7.9
<code>IsDebugMode()</code>	Возвращает <code>True</code> , если в системе включен режим отладки	7.10
<code>IsDefined("variable_name")</code>	Возвращает <code>True</code> , если переменная существует	7.11
<code>IsLeapYear(year)</code>	Возвращает <code>True</code> , если указанный год является високосным	7.12
<code>IsNumeric(string)</code>	Возвращает <code>True</code> , если указанное значение является числовым	7.13
<code>IsNumericDate(number)</code>	Возвращает <code>True</code> , если указанное значение является реальным значением объекта формата даты	7.14
<code>IsProtected(resource_type, resource_name[, action])</code>	Возвращает <code>True</code> , если ресурс защищен в контексте безопасности авторизованного пользователя	7.15
<code>IsQuery(value)</code>	Возвращает <code>True</code> , если значение является запросом	7.16
<code>IsSimpleValue(value)</code>	Возвращает <code>True</code> , если значение является строкой, числом, логической переменной или датой	7.17
<code>IsStruct(variable)</code>	Возвращает <code>True</code> , если значение является структурой	7.18
<code>IsWDDX(value)</code>	Возвращает <code>True</code> , если значение является хорошо сформированным пакетом WDDX	7.19
<code>LSIsCurrency(string)</code>	Возвращает <code>True</code> , если строка является денежной единицей в виде валюты	7.20
<code>LSIsDate(string)</code>	Возвращает <code>True</code> , если указанную строку можно преобразовать, используя одну из функций обработки дат (подобно функции <code>IsDate</code>)	7.21

(оконание)

Функция	Описание	Код
LSIsNumeric(string)	Возвращает True, если указанное значение является числовым (подобно функции IsNumeric)	7.22
ParameterExists(parameter)	Выполняет проверку существования параметра и при обнаружении возвращает True	7.23

Функции управления запросами

Функция	Описание	Код
QueryAddColumn(query, column-name, array-name)	Добавляет новый столбец в запрос и выполняет этот запрос	8.1
QueryAddRow(query [, number])	Добавляет новую строку в запрос	8.2
QueryNew(columnlist)	Создает новый объект запроса с определением списка столбцов	8.3
QuerySetCell(query, column_name, value [, row_number])	Устанавливает ячейку в заданном столбце на заданное значение	8.4

Функции управления массивами

Функция	Описание	Код
ArrayAppend(array, value)	Добавляет элемент в конец заданного массива	9.1
ArrayAvg(array)	Возвращает среднее значение по всем элементам массива	9.2
ArrayClear(array)	Удаляет все данные из массива	9.3
ArrayDeleteAt(array, position)	Удаляет элемент массива, находящийся на указанной позиции	9.4
ArrayInsertAt(array, position, value)	Вставляет элемент массива на указанную позицию	9.5
ArrayIsEmpty(array)	Проверяет, является ли массив пустым	9.6
ArrayLen(array)	Возвращает длину заданного массива	9.7

(окончание)

Функция	Описание	Код
<code>ArrayMax(array)</code>	Возвращает максимальное значение массива	9.8
<code>ArrayMin(array)</code>	Возвращает минимальное значение массива	9.9
<code>ArrayNew(dimension)</code>	Создает массив размерностью от 1 до 3	9.10
<code>ArrayPrepend(array, value)</code>	Добавляет элемент в начало заданного массива	9.11
<code>ArrayResize(array, minimum_size)</code>	Определяет размер массива	9.12
<code>ArraySet(array, start_pos, end_pos, value)</code>	Присваивает определенное значение ряду элементов одномерного массива в указанном диапазоне	9.13
<code>ArraySort(array, sort_type [, sort_order])</code>	Сортирует элементы массива	9.14
<code>ArraySum(array)</code>	Возвращает сумму значений массива	9.15
<code>ArraySwap(array, position1, position2)</code>	Переставляет местами элементы массива	9.16
<code>ArrayToList(array [, delimiter])</code>	Возвращает список элементов, преобразуя данные из массива	9.17

Функции управления списками

Функция	Описание	Код
<code>ListAppend(list, value [, delimiters])</code>	Добавляет значение в конец списка	10.1
<code>ListChangeDelims(list, new_delimiter [, delimiters])</code>	Заменяет в списке все разделители	10.2
<code>ListContains(list, substring [, delimiters])</code>	Возвращает позицию в списке, начиная с которой встречается заданный фрагмент текста (подстрока)	10.3
<code>ListContainsNoCase(list, substring [, delimiters])</code>	Возвращает позицию в списке, начиная с которой встречается заданный фрагмент текста без учета регистра	10.4

(продолжение)

Функция	Описание	Код
ListDeleteAt(list, position [, delimiters])	Удаляет значение из заданной позиции	10.5
ListFind(list, value [, delimiters])	Осуществляет поиск значений в списке с заданным разделителем	10.6
ListFindNoCase(list, value [, delimiters])	Выполняет поиск значений в списке без учета регистра	10.7
ListFirst(list [, delimiters])	Возвращает первый элемент списка	10.8
ListGetAt(list, position [, delimiters])	Возвращает элемент из заданной позиции в списке	10.9
ListInsertAt(list, position, value [, delimiters])	Вставляет значение в заданную позицию списка	10.10
ListLast(list [, delimiters])	Возвращает последний элемент списка	10.11
ListLen(list [, delimiters])	Возвращает длину списка	10.12
ListPrepend(list, value [, delimiters])	Добавляет значение в начало списка	10.13
ListQualify(list, qualifier [, delimiters] [, elements])	Выделяет элементы списка с помощью указанного символа или набора символов	10.14
ListRest(list [, delimiters])	Возвращает список без первого значения	10.15
ListSetAt(list, position, value [, delimiters])	Заменяет значение в заданной позиции списка	10.16
ListSort(list, sort_type [, sort_order] [, delimiter])	Сортирует элементы списка с учетом разделителя	10.17
ListToArray(list [, delimiter])	Возвращает массив, преобразуя данные из списка с учетом указанного разделителя	10.18
ListValueCount(list, value [, delimiters])	Возвращает количество вхождений заданного значения в список	10.19
ListValueCountNoCase(list, value [, delimiters])	Возвращает количество вхождений заданного значения в список без учета регистра	10.20
QuotedValueList(query.column [, delimiter])	Возвращает список значений колонки предварительно выполненного запроса с выделением элементов списка с помощью кавычек	10.21

(окончание)

Функция	Описание	Код
<code>ValueList(query.column [, delimiter])</code>	Возвращает список значений колонки предварительно выполненного запроса	10.22

Функции управления структурами

Функция	Описание	Код
<code>StructAppend(struct1, struct2, overwriteFlag)</code>	Добавляет в конец одной структуры другую структуру	11.1
<code>StructClear(structure)</code>	Удаляет все данные из структуры	11.2
<code>StructCopy(structure)</code>	Копирует структуру	11.3
<code>StructCount(structure)</code>	Возвращает количество ключей в заданной структуре	11.4
<code>StructDelete(structure, key [, indicatenotexisting])</code>	Удаляет элемент структуры с указанным значением ключа	11.5
<code>StructFind(structure, key)</code>	Выполняет поиск данных по ключу в указанной структуре	11.6
<code>StructFindKey(top, value, scope)</code>	Выполняет поиск в подструктуре вложенной структуры и возвращает ряд, содержащий какие-либо структуры со значением, отвечающим заданному параметру	11.7
<code>StructFindValue(top, value [, scope])</code>	Выполняет поиск в подструктуре вложенной структуры и возвращает ряд, содержащий какие-либо структуры со значением, отвечающим заданному параметру	11.8
<code>StructGet(pathDesired)</code>	Возвращает порядок структуры из указанного пути	11.9
<code>StructInsert(structure, key, value [, allowoverwrite])</code>	Вставляет пару "ключ/значение" в заданную структуру	11.10
<code>StructIsEmpty(structure)</code>	Проверяет, является ли структура пустой	11.11
<code>StructKeyArray(structure)</code>	Возвращает массив ключей, преобразуя данные из структуры	11.12
<code>StructKeyExists(structure, key)</code>	Проверяет наличие ключа в массиве	11.13

(окончание)

Функция	Описание	Код
StructKeyList(structure, [delimiter])	Возвращает список ключей, преобразуя данные из структуры	11.14
StructNew()	Создает новую структуру	11.15
StructSort(base, pathToSubElement, sortOrder, sortType)	Выводит ряд структур, содержащих ключевые имена верхнего уровня, отсортированными в соответствии с величиной заданного подэлемента	11.16
StructUpdate(structure, key, value)	Производит обновление данных в структуре	11.17

Функции форматирования

Функция	Описание	Код
DateFormat(date [, mask])	Возвращает отформатированное значение даты и времени	12.1
DecimalFormat(number)	Возвращает результат в десятичном формате	12.2
DollarFormat(number)	Возвращает число в долларовом формате	12.3
FormatBaseN(number, radix)	Преобразует число в строку с заданным основанием системы счисления	12.4
HTMLCodeFormat(string [, version])	Возвращает HTML-строку с удаленными специальными символами между парой тегов <PRE></PRE>	12.5
HTMLEditFormat(string [, version])	Возвращает HTML-строку с удаленными специальными символами	12.6
LSCurrencyFormat(number [, type])	Форматирует значение валюты в зависимости от типа местной валюты	12.7
LSDateFormat(date [, mask])	Форматирует данные даты	12.8
LSEuroCurrencyFormat(currency [, type])	Форматирует значение еврова­люты в зависимости от типа ме­стной валюты	12.9

(окончание)

Функция	Описание	Код
<code>LSNumberFormat(number [, mask])</code>	Форматирует число, используя местное соглашение	12.10
<code>LSTimeFormat(time [, mask])</code>	Форматирует данные времени	12.11
<code>NumberFormat(number [, mask])</code>	Форматирует число	12.12
<code>ParagraphFormat(string)</code>	Форматирует строку, преобразуя ее в абзац	12.13
<code>TimeFormat(time [, mask])</code>	Форматирует данные времени	12.14
<code>YesNoFormat(value)</code>	Возвращает логическое значение или число к виду <i>Yes</i> или <i>No</i>	12.15

Другие функции

Функция	Описание	Код
<code>CreateObject("COM", class, context, serverName "CORBA", class, context, locale "JAVA", class)</code>	Создает COM-, CORBA- или JAVA-объект	13.1
<code>CreateUUID()</code>	Создает универсальный идентификатор с уникальным значением	13.2
<code>DE(string)</code>	Возвращает значение переменной, окруженное двойными кавычками	13.3
<code>DeleteClientVariable("name")</code>	Принудительно удаляет клиентскую переменную	13.4
<code>Duplicate(variable_name)</code>	Возвращает дубликат переменной	13.5
<code>Evaluate(string_expression1 [, string_expression2 [, ...]])</code>	Оценивает строковые аргументы слева направо и возвращает результат оценки	13.6
<code>GetBaseTagData(tagname [, instancenumber])</code>	Возвращает объект, который содержит данные из порожденного тега	13.7
<code>GetBaseTagList()</code>	Возвращает список порожденных тегов	13.8
<code>GetBaseTemplatePath()</code>	Возвращает путь к базовой странице шаблона	13.9
<code>GetClientVariablesList()</code>	Возвращает список клиентских переменных	13.10

(окончание)

Функция	Описание	Код
<code>Iif(condition, string_expression1, string_expression2)</code>	Обрабатывает одно из выражений в зависимости от выполнения условия <code>condition</code>	13.11
<code>PreserveSingleQuotes(variable)</code>	Предотвращает автоматическое выведение одиночных кавычек из переменной	13.12
<code>SetLocale(new_locale)</code>	Устанавливает версию локализации языка для текущей сессии	13.13
<code>SetVariable(name, value)</code>	Устанавливает новое значение для переменной	13.14
<code>StripCR(string)</code>	Возвращает строку с удаленными символами возврата каретки	13.15
<code>WriteOutput(string)</code>	Выводит на экран строку символов	13.16



Теги CFML

Чтобы облегчить поиск и восприятие информации, данное приложение разделено на две части: в первой части теги сгруппированы в алфавитном порядке (без учета первых двух символов CF) с указанием условного кода, по которому можно найти описание того или иного тега во второй части данного приложения, где они сгруппированы в порядке принадлежности к той или иной категории с кратким описанием синтаксиса.

Условный код представляет собой два числа, разделенных точкой, где первое число указывает на номер раздела принадлежности тега, а второе число представляет собой порядковый номер тега внутри раздела.

Например, тег <CFABORT>, имеющий код 7.1, предполагает, что описание данного тега следует искать в седьмом разделе, а сам тег описан первым по порядку.

Алфавитный порядок

A

<CFABORT>	7.1	<CFASSOCIATE>	10.2
<CFAPPLET>	1.1	<CFAUTHENTICATE>	10.3
<CFAPPLICATION>	10.1		

B

<CFBREAK>	7.2

C

<CFCACHE>	11.1	<CFCONTENT>	4.2
<CFCOL>	4.1	<CFCOOKIE>	9.1
<CFCOLLECTION>	6.1		

D

<CFDIRECTORY>	3.1		<CFDUMP>	9.2

E

<CFERROR>	5.1		<CFEXIT>	7.3
<CFEXECUTE>	6.2			

F

<CFFILE>	3.2		<CFFORM>	1.2
<CFFLUSH>	11.2		<CFFTP>	8.1

G

<CFGRAPH>	6.3		<CFGRIDCOLUMN>	1.4
<CFGRAPHDATA>	6.4		<CFGRIDROW>	1.5
<CFGRID>	1.3		<CFGRIDUPDATE>	1.6

H

<CFHEADER>	11.3		<CFHTTP>	8.2
<CFHTMLHEAD>	11.4		<CFHTTTPARAM>	8.3

I

<CFIF>/<CFELSEIF>/<CFELSE>	7.4		<CFINDEX>	6.5
<CFIMPERSONATE>	10.4		<CFINPUT>	1.7
<CFINCLUDE>	11.5		<CFINSERT>	2.1

L

<CFLDAP>	8.4		<CFLOG>	4.3
<CFLOCATION>	7.5		<CFLOOP>	7.6
<CFLOCK>	10.5			

M

<CFMAIL>	8.5		<CFMODULE>	10.6
<CFMAILPARAM>	8.6			

O

<CFOBJECT>	6.6	<CFOUTPUT>	4.4

P

<CFPARAM>	9.3	<CFPROCPARAM>	2.2
<CFPOP>	8.7	<CFPROCRESULT>	2.3
<CFPROCESSINGDIRECTIVE>	4.5		

Q

<CFQUERY>	2.4	<CFQUERYPARAM>	2.5

R

<CFREGISTRY>	9.4	<CFRETHROW>	5.2
<CFREPORT>	6.7		

S

<CFSAVECONTENT>	9.5	<CFSET>	9.7
<CFSCHEDULE>	9.6	<CFSETTING>	11.6
<CFSCRIPT>	10.7	<CFSILENT>	11.7
<CFSEARCH>	6.8	<CFSLIDER>	1.9
<CFSELECT>	1.8	<CFSTOREDPROC>	2.6
<CFSERVLET>	6.9	<CFSWITCH>/<CFCASE>/ <CFDEFAULTCASE>	7.7
<CFSERVLETPARAM>	6.10		

T

<CFTABLE>	4.6	<CFTREE>	1.11
<CFTEXTINPUT>	1.10	<CFTREEITEM>	1.12
<CFTHROW>	5.3	<CFTRY>/<CFCATCH>	5.4
<CFTRANSACTION>	2.7		

U

<CFUPDATE>

2.8

W

<CFWDDX>

6.11

Категории тегов

Будем использовать следующую классификацию тегов:

1. Теги построения форм.
2. Теги управления базами данных.
3. Теги управления файлами и каталогами.
4. Теги управления выводом.
5. Теги обработки ошибок.
6. Теги расширения функциональности.
7. Теги контроля за процессом исполнения страниц.
8. Теги Internet-протоколов.
9. Теги манипулирования переменными.
10. Теги управления приложениями.
11. Другие теги.

Теги построения форм

Тег	Описание	Код
<CFAPPLET>	Вставляет апплет внутрь тега <CFFORM>	1.1
<CFFORM>	Строит форму ввода	1.2
<CFGRID>	Используется с тегом <CFFORM> для создания процедуры управления табличными данными	1.3
<CFGRIDCOLUMN>	Используется с тегом <CFGRID> внутри тега <CFFORM>, позволяя определить столбцы	1.4
<CFGRIDROW>	Используется с тегом <CFGRID> внутри тега <CFFORM>, позволяя заполнять таблицу данными при отказе от использования запроса в качестве источника данных	1.5

(окончание)

Тег	Описание	Код
<CFGRIDUPDATE>	Выполняет обновления таблицы данных	1.6
<CFINPUT>	Используется с тегом <CFFORM> для создания элемента ввода	1.7
<CFSELECT>	Используется с тегом <CFFORM> для создания раскрывающего списка	1.8
<CFSLIDER>	Используется с тегом <CFFORM> для создания движка, перемещаемого по вертикали или горизонтали и позволяющего визуальнo управлять числовыми величинами в указанном диапазоне	1.9
<CFTEXTINPUT>	Используется с тегом <CFFORM> для создания текстового поля ввода	1.10
<CFTREE>	Используется с тегом <CFFORM> для создания дерева иерархического списка	1.11
<CFTREEITEM>	Используется с тегом <CFTREE> внутри тега <CFFORM>, позволяя определить элементы иерархического списка	1.12

Теги управления базами данных

Тег	Описание	Код
<CFINSERT>	Вставляет данные в базу данных	2.1
<CFPROCPARAM>	Задаёт параметры для хранимых процедур в базе данных	2.2
<CFPROCRESULT>	Описывает набор имен, которые другие теги могут использовать для доступа к результатам работы хранимых процедур	2.3
<CFQUERY>	Передаёт на выполнение SQL-запрос в базу данных	2.4
<CFQUERYPARAM>	Используется с тегом <CFQUERY>, проверяя тип данных параметра запроса	2.5
<CFSTOREDPROC>	Определяет информацию соединения с базой данных и задаёт хранимую процедуру, которая должна выполняться	2.6
<CFTRANSACTION>	Группирует запросы в единый пакет транзакции, задавая правила отката в случае ошибок	2.7
<CFUPDATE>	Обновляет информацию в базе данных	2.8

Теги управления файлами и каталогами

Тег	Описание	Код
<CFDIRECTORY>	Осуществляет управление каталогами	3.1
<CFFILE>	Осуществляет управление файлами	3.2

Теги управления выводом

Тег	Описание	Код
<CFCOL>	Определяет характеристики колонки таблицы, созданной с помощью тега <CFTABLE>	4.1
<CFCONTENT>	Определяет тип содержания и имя файла, который должен быть загружен в браузер	4.2
<CFLOG>	Позволяет записывать сообщения в log-файл	4.3
<CFOUTPUT>	Позволяет выводить значения различных переменных CFML, результатов запросов и других необходимых фрагментов текстов	4.4
<CFPROCESSINGDIRECTIVE>	Удаляет лишние пробелы внутри CFML-конструкций	4.5
<CFTABLE>	Позволяет строить таблицу на стороне ColdFusion	4.6

Теги обработки ошибок

Тег	Описание	Код
<CFERROR>	При возникновении ошибок показывает специально созданные страницы для каждого типа ошибок	5.1
<CFRETHROW>	Если вам необходимо выполнить некоторые действия при возникновении исключения во внутреннем блоке <CFTRY>, но не прекратить на этом обработку, а передать исключение дальше, внешнему обработчику, вы можете воспользоваться тегом <CFRETHROW>, позволяющим регенерировать исключение	5.2
<CFTHROW>	Активизирует обработку исключительной ситуации на странице, которая затем может быть отработана с помощью тега <CFCATCH>	5.3

(окончание)

Тег	Описание	Код
<CFTRY>/<CFCATCH>	Позволяет отлаживать и обрабатывать исключительные ситуации во время работы приложения	5.4

Теги расширения функциональности

Тег	Описание	Код
<CFCOLLECTION>	Создает и администрирует коллекцию, используется при работе с поисковым сервером Verity	6.1
<CFEXECUTE>	Запускает выполнение любого процесса на сервере	6.2
<CFGRAPH>	Используется для построения диаграмм, требуется наличия пакета Java-приложений JRun 3.0 Server	6.3
<CFGRAPHDATA>	Используется с тегом <CFGRAPH> для определения данных при построении диаграммы	6.4
<CFINDEX>	Используется для создания индексов коллекции в операциях поиска <CFSEARCH>	6.5
<CFOBJECT>	Вызывает объекты COM, CORBA, Java или EJB	6.6
<CFREPORT>	Запускает предопределенный отчет, созданный в Crystal Reports	6.7
<CFSEARCH>	Выполняет поиск по данным, индексированным в коллекции с помощью тега <CFINDEX>	6.8
<CFSERVLET>	Выполняет Java-сервлет с использованием средств сервера Java-приложений JRun 3.0 Server	6.9
<CFSERVLETPARAM>	Используется с тегом <CFSERVLET> для определения передаваемых параметров	6.10
<CFWDDX>	Упаковывает и распаковывает структуры CFML-данных для WDDX-формата	6.11

Теги контроля за процессом исполнения страниц

Тег	Описание	Код
<CFABORT>	Останавливает обработку CFML-страницы в месте расположения тега	7.1
<CFBREAK>	Используется для прерывания работы цикла, организованного конструкцией <CFLOOP>	7.2
<CFEXIT>	Прерывание обработки тега	7.3
<CFIF>/<CFELSEIF>/<CFELSE>	Условный оператор, обеспечивающий переход к фрагменту кода в зависимости от заданного условия	7.4
<CFLOCATION>	Открывает CFML- или HTML-страницу	7.5
<CFLOOP>	Обеспечивает цикл повторений, основываясь на наборе заданных условий	7.6
<CFSWITCH>/<CFCASE>/<CFDEFAULTCASE>	Оператор-переключатель, обеспечивающий переключение между различными частями программного кода в зависимости от значения одной переменной или выражения	7.7

Теги Internet-протоколов

Тег	Описание	Код
<CFFTP>	Предоставляет возможность работы с файловой системой на стороне определяемого FTP-сервера	8.1
<CFHTTP>	Используется для выполнения методов GET и POST для загрузки или отправки формы, запроса, cookies или CGI-переменной на заданный сервер	8.2
<CFHTTPPARAM>	Применяется вместе с тегом <CFHTTP> для задания передаваемых параметров при использовании метода POST	8.3
<CFLDAP>	Обеспечивает доступ к серверу LDAP (Lightweight Directory Access Protocol)	8.4
<CFMAIL>	Позволяет отправлять почтовые сообщения	8.5
<CFMAILPARAM>	Используется с тегом <CFMAIL>, позволяет либо присоединять файл, либо добавлять заголовки к сообщению	8.6

(окончание)

Тег	Описание	Код
<CFPOP>	Позволяет извлекать сообщения из почтового сервера	8.7

Теги манипулирования переменными

Тег	Описание	Код
<CFCOOKIE>	Определяет и устанавливает переменные cookie	9.1
<CFDUMP>	Позволяет наглядно демонстрировать значения простых переменных, запросов, массивов, структур и WDDX-переменных	9.2
<CFPARAM>	Определяет переменные и их значения по умолчанию	9.3
<CFREGISTRY>	Позволяет считывать, записывать и удалять ключи и их значения в системном реестре операционной системы	9.4
<CFSAVECONTENT>	Сохраняет содержание тега тела страницы <BODY> в определяемую переменную	9.5
<CFSCHEDULE>	Назначает задачу по выполнению конкретных страниц с возможностью генерации статистических документов	9.6
<CFSET>	Определяет переменные ColdFusion	9.7

Теги управления приложениями

Тег	Описание	Код
<CFAPPLICATION>	Используется для определения основных характеристик приложения	10.1
<CFASSOCIATE>	Позволяет данным порожденных тегов храниться вместе с информацией базового тега	10.2
<CFAUTHENTICATE>	Идентифицирует пользователя для предоставления доступа к защищенному содержанию приложения	10.3
<CFIMPERSONATE>	Позволяет на короткое время предоставить доступ к защищенному содержанию, тем самым, оказать пользователю доверительное отношение	10.4

(окончание)

Тег	Описание	Код
<CFLOCK>	Синхронизирует раздел CFML-кода, используя блокировку для обеспечения целостности разделяемых данных	10.5
<CFMODULE>	Используется для вызова специального тега	10.6
<CFSCRIPT>	Используется для определения границ кода, написанного на языке CFScript	10.7

Другие теги

Тег	Описание	Код
<CFCACHE>	Создает временные файлы на основе страниц ColdFusion, которые содержат статические HTML-страницы, позволяя тем самым ускорить отображение страниц	11.1
<CFFLUSH>	Направляет потоки данных клиенту с определенным интервалом по мере их изменения	11.2
<CFHEADER>	Генерирует HTTP-заголовки, которые возвращаются клиенту	11.3
<CFHTMLHEAD>	Пишет текст, указанный атрибутом <code>TEXT</code> , в секцию <code><HEAD></code> генерируемой страницы	11.4
<CFINCLUDE>	Включает указанный шаблон в текущую страницу	11.5
<CFSETTING>	Определяет и управляет различными установками в ColdFusion	11.6
<CFSILENT>	Останавливает все выводы, порожденные выполнением CFML-конструкций внутри области действия тега	11.7